edv aspekte

1/86

Herausgegeben
von der Redaktion
rechentechnik
datenverarbeitung
DDR 5.00 M

Betriebssysteme für Bürocomputer

A 5120 A 5130
A 5120.16

Inhalt

Volkmar Köhler:
Überblick über Hardware und Software der Bürocomputer 2

Barbara Bündig, Otbert Fröhlich, Heinz-Holger Lange, Peter Ulbrich:
Das Betriebssystem MUTOS 8000 5

Volkmar Köhler, Hans-Gerd Unterschütz:
Das Betriebssystem SIOS 17

Volkmar Köhler:
Das Betriebssystem UDOS 31

Hans-Gerd Unterschütz:
Das Betriebssystem SCP 46

Содержание

Аппаратная часть и матобеспечение для конторских компьютеров. Обзор 2

Барбара Бюндиг, Отберт Фрелих, Хайнц-Хольгер Ланге, Петер Ульбрих:
Операционная система МУТОС 8000 5

Фолькмар Келер, Ханс-Герд Унтершюц:
Операционная система СИОС 17

Фолькмар Келер:
Операционная система УДОС 31

Ханс-Герд Унтершюц;
Операционная система SCP 46

In this Issue

Volkmar Köhler:
Survey of office computer hardware and software 2

Barbara Bündig, Otbert Fröhlich, Heinz-Holger Lange, Peter Ulbrich:
The MUTOS 8000 operating system 5

Volkmar Köhler, Hans-Gerd Unterschütz:
The SIOS operating system 17

Volkmar Köhler:
The UDOS operating system 31

Hans-Gerd Unterschütz:
The SCP operating system 46

Jahresinhaltsverzeichnis 1985

1/85
Programmierung von Industrierobotern

Henning Zachau: Automatisierung komplexer Systeme 3

Jan Schwartz: Stand und Trend bei Roboterprogrammiersprachen 6

György Varga, Michael Krapp: FORTH als Sprachkonzept für Robotersteuerungen 11

Joachim Hübener: Das modulare Robotersteuerungs- und Programmiersystem ARC 16

Joachim Hübener, Adreas Moltmann: Programmierkomponenten für das Softwaresystem ARC 20

Jürgen von Pistor, Tobias Heim: Steuerung flexibler automatisierter Fertigungszellen 24

Wilfried Kreisel: Einsatz von Sensoren in Robotersystemen 28

Helge Herz: Arithmetik in Robotersteuerungen 33

Ingeborg Reddmann: Positionierfehler an Gelenkrobotern 36

Harald Loose, Claus-Dieter Schmidt: Optimierung der Bewegungsabläufe von Industrierobotern 38

Michael Weingart: Das Betriebssystem des Industrieroboters PHM 41 42

Lutz Jakubik: Die Steuerung des Montageroboters ZMR 1 46

Siegfried Prinz: Anwenderprogrammierung und -generierung der IRS 650 49

Siegfried Prinz, Rainer Schmidt: Sensoreinsatz für die Bahnsteuerung IRS 650 51

Gernot Meyer: Industrierobotersteuerungssystem IRS 700 53

Peter Haferkorn: Direkte Teach-In-Programmierung für Farbspritzroboter 54

Rolf Breitbarth: Bedienarme Wälzschraubtriebproduktion mit dem ZIM 60-1 55

Klaus-Jürgen Peter, Rainer Kirsten: Technologische Einheit mit ZIM 60-1 57

Detlef Wartensleben: Einsatz des ZIM 60-1 zum Sandkiesstrahlen 62

2/85
CAD/CAM
Entwicklung und Anwendung

Johannes Klose, Detlef Kochan: Durchgängige automatisierte Systeme 2

Jochen Winkler: Der AKT A 6454 und seine Weiterentwicklung 5

Peter Fischer, Richard Krahmer: Rastersichtgerät K 8917 und Tendenzen der Weiterentwicklung 8

Karl-Heinz Kutschke, Erhard Berndt: Nutzer-Maschine-Kommunikation in CAD-Systemen 11

Klaus Riedel, Herbert Schreiter: Realisierung eines CAD-Rahmensystems 13

Bernd Kehrer, Hans-Steffen Grosch: Datenbehandlung in CAD-Systemen 15

Adolf Kotzauer: Standardisierte Grafik-Software 18

Heinz-W. Eberl, Michael Polnick, Hartmut Anke: Basiselement GKS für CAD/CAM-Arbeitsplätze 22

Manfred Ludwig, Christine Richter: Geometriemoduln für CAE-Systeme 25

Andreas Hartwig, Klaus Mager, Karla Nestler: Das 3D-Modellierungssystem GEKO 28

Horst Weigert: Modellierung technischer Objekte 30

Detlef Kochan, Dieter Fichtner: Entwurf und Fertigung doppelt gekrümmter Flächen 32

Gert Bär: Ein CAD/CAM-System für Schnekken und Gewinde 35

Ekkehart Pippig, Bernd Rietschel, Herbert Schreiter: Skulpturartige Flächen im System 3K 37

Gerhard Kretzschmar, Peter Eisold, Helmar Lieber, Stefan Zentgraf, Rolf Zimmermann: CAD zur Entwicklung von Werkzeugmaschinengestellen 39

Friedmar Erfurt: CAD-orientierte Verfahren der Finite-Elemente-Methode 45

Horst Kretzschmar: CAD-System zur Bausteinprojektierung im Wohnungsbau 47

Horst Schmidt: TOPOLOGIE – Subsystem zur Erzeugung von Strukturen 51

Horst Elschner, Albrecht Möschwitzer, Albrecht Reibiger: CAD/CAM-Systeme für Schaltkreisentwicklung und -fertigung 52

Frank Hoffmann: Einsatz automatisierter kartographischer Systeme 56

Henry Stahl, Dieter Orlamünder, Bernd Hetze: Computergrafik-Ausbildung für das Maschineningenieurwesen 60

Friedmar Erfurt, Rudi Mauroschat: Computereinsatz in der Fachausbildung von Ingenieuren 63

3/85
Systemprogramme/Programmiersysteme

Erika Horn: Zur Rechnerstützung des Softwareentwurfs 2

Uwe Petermann: Axiomatische Schnittstellenspezifikation eines Dateibehandlungssystems 4

Axel Voigt, Wolfgang Schubert: Datenbasis für ein Softwarekonstruktionssystem 6

Arne Fellien: Speichertechnik für Interpretation von Datenbasen 7

Rüdiger Blach: Implementationen von Modula-2 für 8- und 16-Bit-Prozessoren 9

Helmut Mönch: Ein Modula-2-Compiler für ESER 11

Bodo Hohberg: Editor mit Syntaxorientierung für PASCAL-Programme 14

Stanimir Stojanow: Optimierung von Überlagerungsstrukturen für PASCAL 1600 16

Roland Maertz: Dumpsystem PANDORA für PASCAL 1600 18

Christian Hanisch: Programmverbindung von Sprachen mittels gemischter Batch-Compilation 20

Alfred Schilling: Das Compiler-Writing-System 24

Horst Rothe: Programmentwicklung mit den Sprachen AID und DIPRO 28

Werner Aßmann: Transformationsgrammatiken zur Sprachtransformation im INDIA-System 30

Rainer Staudte: Ein PROLOG-Interpreter für SCP 32

Ulrich Geske: Funktionales und relationales Programmieren in PROLOG 34

Matthias Ehrig, Günther Fischer: Zur Weiterentwicklung des ESER-C-Compilers 37

Horst Sobotta: Realisierung einer Sitzungssteuerung auf Bürocomputern 39

Eckhard Einert: Terminalbetrieb auf universellen Mikrorechnern 41

Dr. Karin Porst, Uwe Frank: Softwarewerkzeuge für Bürocomputer 44

Karsten Kluge: Textverarbeitung mit TEPROS/SKR 46

Niklaus Wirth: MODULA-2 48

4/85
PASCAL – Anwendungen für Wirtschaft, Wissenschaft, Technik

Sammlung von 101 Anwendungsbeispielen mit Quelltexten, Resultaten und Kommentaren

edv aspekte

5. Jahrgang 1/1986

Verlag Die Wirtschaft Berlin
1055 Berlin, Am Friedrichshain 22
Verlagsdirektor: Dieter Grüneberg

edv-aspekte
Zeitschrift für spezielle Themen der Informationsverarbeitung, herausgegeben von der Redaktion rechentechnik/datenverarbeitung, 1055 Berlin, Am Friedrichshain 22
Chefredakteur: Franz Loll 4 38 73 41
Redakteur: Hans Weiß 4 38 73 16
Sekretariat: 4 38 72 33
Fernschreiber: 114 566
Titelgestaltung: Marlies Hawemann

Redaktionsschluß: 29. Oktober 1985

Lizenz des Presseamtes beim Vorsitzenden des Ministerrates der DDR Nr. 1529

edv-aspekte
Erscheinungsweise vierteljährlich zum Bezugspreis DDR 5,00 M je Heft
EDV-Artikel-Nr. 1331
Auslandspreise sind dem Zeitschriftenkatalog des Außenhandelsbetriebes Buchexport zu entnehmen.

Satz: Verlag Die Wirtschaft, Berlin
Druck: (140) „Neues Deutschland", Berlin

Anzeigenverwaltung:
Berliner Verlag, 1056 Berlin, Karl-Liebknecht-Str. 29, Telefon: 2 70 33 02

Anzeigenannahme:
Berliner Verlag und Annahmestellen in Berlin und in den Bezirken
Zur Zeit gültige Anzeigenpreisliste Nr. 12

Im Ausland:
INTERWERBUNG GmbH – Gesellschaft für Werbung und Auslandsmessen der DDR, 1157 Berlin, Hermann-Duncker-Str. 89

Bestellungen nehmen entgegen:
Für die DDR
Sämtliche Postämter, der örtliche Buchhandel und der Verlag Die Wirtschaft Berlin
Inkasso-Zeitraum: vierteljährlich

Im Ausland:
In den sozialistischen Ländern nur der zuständige Postzeitungsvertrieb. In allen anderen Staaten der örtliche Buch- und Zeitschriftenhandel. Bestellungen des Buch- und Zeitschriftenhandels sind zu richten an

BUCHEXPORT
Volkseigener Außenhandelsbetrieb der DDR, DDR – 7010 Leipzig, Leninstr. 16, Postfach 160
oder an Verlag Die Wirtschaft, Berlin
DDR – 1055 Berlin, Am Friedrichshain 22

Mitglieder des Redaktionsbeirates
Dr. Claus Goedecke · Dr. Rolf Gräßler
Prof. Dr. sc. Gerhard Keßler · Dr. Rolf Kilian
Hans Kunau · Walter Münch · Axel Rathsack
Prof. Dr. sc. Wolfgang Schoppan (Vorsitzender)
Dr. Werner Schulze · Horst Stoll
Prof. Dr. Franz Stuchlik · Dr. Dieter Urban

Bürocomputerbetriebssysteme – Schnittstelle zwischen Hardware und Nutzer

Auf der 10. Tagung des Zentralkomitees der SED stellte Genosse Erich Honecker fest, daß die Effizienz der Arbeit der Ingenieure mit dem Einsatz von Büro- und Personalcomputern erheblich gesteigert werden kann. „Routinemäßige Tätigkeit könnte automatisiert, Zeit und Energie für schöpferische Prozesse gewonnen werden. Die Beschleunigung hier würde sich bis in die volkswirtschaftlichen Steigerungsraten der Arbeitsproduktivität fortsetzen". Und weiter: „Logischerweise macht es sich notwendig, die Produktion von Büro- beziehungsweise Personalcomputern wesentlich zu erhöhen. Was wir für diese Rationalisierungsmittel der geistigen Arbeit aufwenden, fließt in kurzer Frist mehrfach zurück." Der VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt unternimmt große Anstrengungen, um diese anspruchsvollen Ziele zu erfüllen.

Die Bürocomputer A 5120 und A 5130 werden seit mehreren Jahren erfolgreich in der Volkswirtschaft der DDR angewendet. Sie bestehen aus einer relativ universellen Grundhardware, die im Verlaufe der Entwicklung und Produktion ständig erweitert und ergänzt wurde. Schritthaltend dazu erfolgte auch die sukzessive Weiterentwicklung der für diese Erzeugnisse erforderlichen Software. Mit dem Angebot mehrerer Betriebssysteme wurde die Anwendungsbreite der Bürocomputer enorm vergrößert, so daß sie heute in fast allen Bereichen der Volkswirtschaft eingesetzt werden. Da die Zahl der Anwender laufend zunimmt, wächst auch das Informationsbedürfnis über Möglichkeiten und Grenzen dieser neuen Technik ständig an.

Ausgehend von den mehrjährigen Erfahrungen der Autoren bei Lehrgängen und Vorlesungen wurde deshalb diese konzentrierte Form der Informationsvermittlung gewählt. Oft ist es so, daß viele Anwender der Bürocomputer das erste Mal direkt mit der Rechentechnik konfrontiert werden. Sie sollen systematisch mit der Betriebssoftware der Bürocomputer vertraut gemacht werden. Andere Anwender arbeiten schon länger mit einem bestimmten Betriebssystem. Ihnen sollen die neuerschlossenen Anwendungsbereiche der zwischenzeitlich eingeführten Betriebssysteme verdeutlicht werden. Denn es erweist sich immer wieder, daß jedes Betriebssystem effektiv nur eine bestimmte Anwendungsbreite abdeckt, in der Praxis aber die Aufgaben weitaus breiter gefächert sind. So wird es vor allem immer dann, wenn mehrere Nutzer sich in die Nutzung eines Bürocomputers teilen, sinnvoll und rationell sein, mit mehreren Betriebssystemen zu arbeiten.

In diesem Heft sind aufeinander abgestimmte Beiträge zu den einzelnen Betriebssystemen enthalten, wobei für jedes Betriebssystem die gleiche Gliederung gewählt wurde, so daß ein Vergleich der Betriebssysteme bei den einzelnen Parametern leicht möglich ist.

Dr. Volkmar Köhler

Überblick über Hardware und Software der Bürocomputer

Dr. Volkmar Köhler
VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt

1. Einführung*

Strenggenommen bedeutet der Begriff *Bürocomputer*, daß es sich hier um Rechentechnik für ausschließlich kommerzielle Zwecke, wie Büroautomatisierung, Datenerfassung, Buchung und Fakturierung u. ä., handelt. Mittlerweile wird dieser Begriff aber wesentlich weiter gefaßt. So soll auch in diesem Beitrag der Begriff Bürocomputer als Synonym für Geräte des unteren bis mittleren Preis/Leistungsbereiches in der Mikrorechentechnik stehen, wobei von diesen Geräten eine variable technische und programmtechnische Ausstattung gefordert wird. Über entsprechende Koppelausstattungen gelingt es mit diesen Geräten, die Leistungen der EDV gewissermaßen an den Arbeitsplatz, wo sie gebraucht werden, zu transportieren.
In der internationalen Literatur findet man für diese rechentechnische Kategorie die Begriffe *Personalcomputer* oder *Arbeitsplatzcomputer* ebenso wie manchmal *universelle* oder *spezielle Terminals*.
Im VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt werden seit mehreren Jahren zwei derartige Grundmodelle, die Bürocomputer A 5120 und A 5130, gefertigt.

1.1. Ausstattungsmöglichkeiten für die Bürocomputer A 5120 und A 5130

Hinsichtlich ihrer Grundausstattung an Hard- und Software kann man die beiden Bürocomputer durchaus als gleichwertig betrachten. Sie sind verschiedene Ausstattungsvarianten des gleichen Systemkonzeptes.
Der BC A 5120 ist ein kompaktes, bildschirmorientiertes Auftischgerät mit abgesetzter Tastatur. Das Grundgefäß enthält den Rechner aus Moduln des Mikrorechnersystems K 1520 mit der ZRE K 2526 und üblicherweise 64 KByte Arbeitsspeicher sowie der Anschlußelektronik für die jeweils angeschlossene Peripherie. Weiterhin sind im Grundgefäß die immer erforderliche Bildschirmbaugruppe, die Stromversorgung für das gesamte Grundgefäß sowie eine Datenträgerbaugruppe untergebracht. Je nach verfügbaren Laufwerken kann letztere aus

– 1 Laufwerk 8″-Floppy-Disk analog MF 3200 bzw. MF 6400
– 1 ... 3 Laufwerken Minifloppy-Disk analog K 5600.10 oder
– 1 ... 2 Laufwerken Kassettenmagnetbandgerät K 5200

bestehen.
Mit dieser Konfiguration ist das System bereits betriebsfähig. Über den Datenträger kann das Betriebssystem geladen werden. Tastatur und Bildschirm sind die Dialoggeräte.
Für viele Zwecke ist ein Drucker unerläßlich. Beim A 5120 kann er ebenfalls als Beistellbaugruppe über ein Kabel angeschlossen werden. Einsetzbar sind der SD 1152 oder SD 1157 mit PIO- oder IFSS-Schnittstelle bzw. die neueren Drucker der Reihe K 6311 ... 16.
Wird eine größere externe Speicherkapazität benötigt, so ist ein Beistellgefäß für Floppy-Disk-Laufwerke anzuschließen. Es enthält dann auch für diese Laufwerke eine eigene Stromversorgung. Anschließbar sind auf diese Weise zwei weitere Laufwerke für 8″-Floppy-Disk, so daß auf einem A 5120 mit maximal drei Standard-Laufwerken gearbeitet werden kann.
Beim Einsatz als Terminal bzw. in Rechnersystemen sind Einrichtungen für die Datenübertragung erforderlich. Die dazu notwendige Anschlußelektronik findet üblicherweise im Grundgefäß Platz.
Für spezielle Einsatzfälle ist die Aufrüstung des Arbeitsspeichers mit einer Speicher-Steckeinheit von 48 KByte für das Betriebssystem SIOS möglich.
Der BC A 5130 ist ein druckerorientiertes Standgerät (kompletter Sitzarbeitsplatz) mit in das Gefäß integrierter Tastatur und Einbaudrucker. Die Bildschirmbaugruppe ist über eine Vorrichtung drehbar gelagert oberhalb des Druckers angebracht. Der Rechner einschließlich Arbeitsspeicher und Stromversorgung ist in einem Anhängegefäß unterhalb des Druckers untergebracht. Die Datenträgerbaugruppe befindet sich als Anhängegefäß oder Beistellschrank für maximale Ausbaustufen links am Grundgefäß. Im Beistellschrank können maximal vier Floppy-Laufwerke enthalten sein.
Auf Grund des leistungsstärkeren Netzteils (gegenüber A 5120, da mehr Platz im Gefäß) können am A 5130 weitere Peripheriebaugruppen mit ihrer Anschlußelektronik installiert werden. Dazu gehören zum Beispiel Lochbandeinheit, $^1/_2$″-Magnetbandeinheit oder ein zweiter Drucker.
Seit 1985 wird für die Bürocomputer eine Rechnererweiterung angeboten. Mittels eines sogenannten Erweiterungsmoduls wird aus dem bisherigen 8-Bit-Bürocomputer ein Rechnersystem mit einem 8-Bit- und einem 16-Bit-Mikrorechner. Basis für den 16-Bit-Mikrorechner ist der Mikroprozessor U 8001. Zu diesem 16-Bit-Rechner gehört eine weitere Speicher-Steckeinheit mit 256 KByte. Die Bezeichnung dieses Bürocomputers lautet A 5120.16.
Auf Grund ihrer modularen Struktur in Hard- und Software ist die Konfigurierbarkeit der Bürocomputer recht groß. Um die Variantenvielfalt nicht unübersichtlich werden zu lassen, befinden sich nur bestimmte Vorzugskonfigurationen im Vertriebsprogramm. Über die konkreten Ausstattungsvarianten geben die zuständigen Absatzbereiche des VEB Kombinat Robotron Auskunft.

1.2. Verfügbare Betriebssysteme

Mit der Einführung der Bürocomputer A 5120 und A 5130 in die Produktion (1980) wurde gleichzeitig ein erstes Betriebssystem (SIOS) in den Vertrieb übernommen. SIOS ist in seiner ursprünglichen Zweckbestimmung hauptsächlich kommerziell orientiert. Es bietet eine maximale Unterstützung für Probleme der

– Datenerfassung und Datenvorverarbeitung
– Abrechnung, Buchung und Fakturierung
– Kommunikation mit anderen Rechnern.

* Da die Beiträge dieses Heftes trotz ihrer Eigenständigkeit einen in sich geschlossenen Gesamtkomplex bilden, wurden die Kapitel durchgehend numeriert.
Red.

Der Anwender erhält die Möglichkeit, seine Probleme in einer eigens für diese Aufgabenklasse geschaffenen Programmiersprache, der Makroassemblersprache MABS, zu formulieren. SIOS ist auch bisher das einzige Betriebssystem geblieben, welches die gesamte in den Bürocomputerbestand eingeführte Hardware betreiben kann. Insbesondere gilt dies für die variable Ausstattung mit externen Speichern (Minifloppy, Standardfloppy, Kassettenmagnetband, 1/2"-Magnetband) und die Datenübertragung.

Mit dem wachsenden Einsatz von Bürocomputern in allen Bereichen der Volkswirtschaft wurde sehr rasch deutlich, daß die Hardware des Bürocomputers weit universeller und leistungsfähiger ist, als sie mit dem Betriebssystem SIOS genutzt werden konnte. Insbesondere für die Programmentwicklung nichtkommerzieller Probleme, wie zum Beispiel für den Einsatz als Programmierarbeitsplatz, war die Nutzung der Bürocomputer mit SIOS nicht befriedigend möglich. Aus diesem Grunde wurde 1983 ein zweites Betriebssystem (UDOS), welches auf diese Aufgabenkategorie zugeschnitten ist, in den Vertrieb eingeführt. Auf Grund seiner Leistungsfähigkeit verbreitete es sich sehr rasch. Sehr wesentlich trugen dazu auch die vier zur Verfügung stehenden höheren Programmiersprachen bei. So werden zum Beispiel heute an nahezu allen Hochschuleinrichtungen der DDR, die den Studenten eine Programmierausbildung bieten, Bürocomputer mit UDOS benutzt.

1984 wurde dann ein drittes Betriebssystem (SCP) für die gleiche Hardware in den Vertrieb übernommen, welches die Anwenderschnittstelle des international eingeführten Betriebssystems CP/M, Version 2.2, realisiert (CP/M ist ein eingetragenes Warenzeichen der Digital Research Corp., USA). Damit war eine Möglichkeit geschaffen, die unter diesem Betriebssystem weltweit vorhandene Anwendersoftware auch auf dem BC A 5120 bzw. BC A 5130 abzuarbeiten. Eines der Hauptanwendungsgebiete ist seither zusammen mit der Nutzung des Textprozessors TEXT 30 die Spezifikation des Bürocomputers als Textverarbeitungssystem.

Parallel mit der Entwicklung des Erweiterungsmoduls für den BC A 5120.16 wurde an der Implementierung eines 16-Bit-Betriebssystems (MUTOS 8000) gearbeitet. MUTOS 8000 nutzt die Ressorcen des 16-Bit-Prozessors U 8001 und des zugehörigen Arbeitsspeichers von 256 KByte. Damit wird der BC A 5120.16 zu einem Programmierarbeitsplatz für den Prozessor U 8000 und schafft gute Voraussetzungen für den Einsatz dieses Prozessors auch in anderen Bereichen der Volkswirtschaft.

1.3. Mittel für die Programmentwicklung

Für jedes der hier vorgestellten Betriebssysteme existieren eine ganze Reihe von Programmierhilfsmitteln (Werkzeugen), um die Erstellung von Programmen schnell und sicher zu ermöglichen.

Bei der Programmentwicklung unterscheidet man zwei grundsätzliche Technologien:

- Assemblertechnologie
- Programmierung mit höheren Programmiersprachen.

Auf die einzelnen Komponenten bei der Nutzung der Assemblertechnologie wird bei jedem Betriebssystem näher eingegangen. Hier soll auch auf das Angebot an höheren Programmiersprachen eingegangen werden.

Für das Betriebssystem SIOS sind höhere Programmiersprachen erst relativ spät realisiert worden. Deshalb ist dort die Tatsache zu verzeichnen, daß kumulativ betrachtet der überwiegende Teil der Anwendersoftware in CPU- oder Makroassemblersprache geschrieben worden ist. Als erste höhere Programmiersprache wurde ein BASIC-Interpreter für SIOS implementiert. Da dieser Interpreter einen ziemlich großen Umfang besitzt (etwa 27 KByte), mußte ein speziell angepaßtes SIOS entwickelt werden, damit beide Komponenten und ein Anwenderprogramm zusammen im Arbeitsspeicher Platz finden. Der für das BASIC-Programm verfügbare Arbeitsspeicher besitzt hier eine durchschnittliche Größe von rund 10 KByte (Wert schwankt in Abhängigkeit von der Hardwarekonfiguration). Da diese Sprache sehr leicht zu erlernen ist und der Interpreter eine recht komfortable Arbeitsweise besitzt, hat das SIOS-BASIC viele Anwender gefunden, trotz des vergleichsweise kleinen Anwenderspeichers.

Später wurden dann noch zwei weitere höhere Sprachen in den Vertrieb genommen:

Zunächst erschien ein „kommerzielles PASCAL", das auf den Anwendungsbereich von SIOS zugeschnittene Sprachelemente enthält. Ähnlich verhält es sich mit dem SIOS-COBOL, das ebenfalls auf kommerzielle Anwendungsbereiche orientiert. Dadurch sind bei beiden Sprachen doch erhebliche Abweichungen zum international standardisierten PASCAL bzw. COBOL zu verzeichnen.

Beim Betriebssystem UDOS wurden von Beginn an vier höhere Programmiersprachen zur Verfügung gestellt:

- BASIC
- PASCAL
- FORTRAN und
- PLZ.

BASIC ist wiederum in Form eines Interpreters implementiert und besitzt den gleichen Sprachvorrat wie SIOS-BASIC. Da die Disketten- und Dateistruktur von SIOS und UDOS aber nicht kompatibel sind, lassen sich BASIC-Programme nicht ohne weiteres von dem einen Betriebssystem zum anderen übertragen.

PASCAL-Programme werden zunächst compiliert. Der Compiler liefert einen sogenannten Zwischenkode, den P-Kode. Dieser wird mit einer weiteren Komponente des PASCAL-Systems verdichtet und anschließend über ein Laufzeitsystem interpretativ abgearbeitet. Diese Technologie hat gegenüber BASIC den Vorteil, daß das Laufzeitsystem, das ebenso wie ein Interpreter bei der Abarbeitung des Anwenderprogrammes speicherresident ist, wesentlich kleiner gehalten werden kann (etwa 12 KByte). Nachteilig bei der PASCAL-Technologie ist, daß kein Programmverbinder existiert, mit dem einzeln übersetzte Module am Schluß zu einem abarbeitungsfähigen P-Kode-Programm

gebunden werden können. So müssen immer alle Programmteile, die zu einem PASCAL-Problem gehören, gemeinsam übersetzt werden.
FORTRAN-Quelltext wird über einen Compiler bis auf einen Objektkode transformiert. Anschließend können mehrere Objektmodule mittels eines speziellen FORTRAN-Linkers zu einem abarbeitungsfähigen Programm gebunden werden. Dieser FORTRAN-Linker ist erforderlich, weil der vom Compiler erzeugte Objektkode von dem des Assemblers abweicht. Die Verwendung des normalen Linkers ist damit nicht möglich. Ebenso ist das Einbinden von Assemblerunterprogrammen in FORTRAN-Programme ohne speziellen Objektkodewandel nicht zu realisieren.
Zum Komplex PLZ gehören zwei Sprachen: PLZSYS und PLZASM. Während PLZSYS als echte höhere Programmiersprache einzuordnen ist, stellt PLZASM nur eine Untermenge davon dar, die mit der Assemblersprache (der Befehle) des jeweiligen Prozessors kombiniert wird. So gibt es ein PLZASM für den 8-Bit-Mikroprozessor, ein weiteres PLZASM für den Einchipmikrorechner und schließlich noch ein PLZASM für den 16-Bit-Mikroprozessor. Gemeinsam sind diesen drei Varianten immer die aus PLZSYS entlehnten Anteile.
Die Verarbeitung von PLZSYS erfolgt compilativ. Es entsteht dabei ein Zwischenkode, der Z-Kode. Dieser kann entweder interpretativ abgearbeitet werden, oder er läßt sich über einen sogenannten Kodegenerator in Objektkode des jeweiligen Prozessors umwandeln. Der Objektkode ist mit dem des entsprechenden Assemblers kompatibel, so daß für Assembler und PLZ auch ein gemeinsamer Linker benutzt werden kann.
Für das dritte 8-Bit-Betriebssystem SCP wurden sukzessive ebenfalls vier höhere Programmiersprachen in den Vertrieb übernommen. Begonnen wurde wieder mit BASIC, wobei diesmal sowohl ein Interpeter als auch ein Compiler zur Verfügung stehen. Der Vorteil liegt darin, daß mit dem Interpreter ein Programm „eingefahren" (getestet) und anschließend compiliert werden kann. Dabei entsteht ein Maschinenprogramm, welches zu seiner Abarbeitung nur noch das Betriebssystem benötigt. Derartige BASIC-Programme können daher weit größer sein, da der Anwenderspeicher nicht durch einen residenten Interpreter teilweise blockiert wird. Allerdings muß hinzugefügt werden, daß das SCP-BASIC in seinem Sprachvorrat geringfügig vom UDOS- bzw. SIOS-BASIC abweicht und bei einer Portierung derartiger Programme auch eine Überarbeitung vorzunehmen ist.
Nach BASIC wurden dann drei weitere höhere Programmiersprachen in den Vertrieb eingeführt: C, PASCAL/M2 und FORTRAN.
Zur Sprache C ist zu bemerken, daß sie vor allem bei 16-Bit-Betriebssystemen immer stärker an Bedeutung gewinnt. Indem unter dem 8-Bit-Betriebssystem SCP bereits die Sprache C zur Verfügung gestellt wurde, vollzog sich ein wichtiger Schritt beim Übergang zur 16-Bit-Mikrorechentechnik in der DDR.
PASCAL/M2 wird über ein Laufzeitsystem abgearbeitet (ähnlich der Arbeitsweise von UDOS-PASCAL). Im Unterschied zu UDOS existiert beim SCP-PASCAL aber ein Linker, wodurch mehrere getrennt compilierte Programme gebunden werden können.
Die Verarbeitung von FORTRAN-Programmen erfolgt analog der Technologie von UDOS-FORTRAN.

1.4.
Ladekonzept

Der Rechner der Bürocomputer ist ein Mikroprozessor (U 880) mit einem Arbeitsspeicher (typisch 64 KByte) und Anschlußsteuerungen. Da die zum Einsatz kommenden Halbleiterbauelemente für den Speicher (dynamische RAM) bei Spannungsabschaltung ihre Informationen verlieren, ist es notwendig, das Betriebssystem bei jedem Neueinschalten des Gerätes vom Datenträger in den Arbeitsspeicher zu laden. Dieser Vorgang läuft automatisch und unabhängig vom Betriebssystem ab, ohne daß Hardware-Änderungen notwendig sind. Das bedeutet, daß der Lader jede Systemdiskette von SIOS, UDOS oder SCP laden muß – ohne Berücksichtigung spezifischer Eigenheiten.
Den eigentlichen Lader kann man so als einen Teil der Hardware (besser: Firmware) betrachten. Bei den Bürocomputern ist dies ein Programm der Länge 1 KByte, das in einem abschaltbaren EPROM auf der ZRE untergebracht ist. Da dieser sogenannte Anfangslader beim Betriebssystem SIOS noch die Sonderfälle

– gemischter RAM/ROM-Arbeitsspeicher
– Systemdatenträger Diskette oder Magnetbandkassette

beachten muß, sind seine Möglichkeiten natürlich begrenzt. Aus diesem Grunde existiert noch ein zweiter Lader, der Systemlader, der sich auf dem Systemdatenträger befindet und als erstes überhaupt geladen wird. Der Systemlader ist bereits in einigen Punkten spezifisch auf das zu ladende Betriebssystem ausgelegt und besitzt die Fähigkeit, den gesamten residenten Teil des jeweiligen Betriebssystems in den Arbeitsspeicher zu laden.

1.4.1.
Anfangslader

Der Anfangslader wird über die Hardware aktiviert, wenn ein RESET-Signal anliegt (z. B. durch Einschalten der Anlage oder zweimaliges Betätigen der Netztaste). Er ist dann adreßmäßig in den niedrigsten Teil des Arbeitsspeichers (Adresse 0 ... 3FFH) eingeordnet. Mittels RESET wird unter anderem ein Sprung auf die Adresse 0 ausgelöst, wodurch der erste Befehl des Anfangsladers abgearbeitet wird.
Der Anfangslader lädt sich zunächst selbst vom ROM in den RAM um (adreßgleiche Speicherzellen). Anschließend wird der ROM durch einen OUT-Befehl auf den auf der ZRE befindlichen System-PIO vom Adreßbereich abgeschaltet. Damit steht nunmehr ein Bereich mit vollen 64 KByte RAM zur Verfügung. Nun wird nur im RAM weitergearbeitet. Der Anfangslader sucht auf allen angeschlossenen Floppy-Disk- und Magnetbandkassetten-Laufwerken nach einem Systemlader. Das erfolgt, indem einfach feste Disketten- bzw.

Das Betriebssystem MUTOS 8000

Dr. Barbara Bündig, Otbert Fröhlich, Heinz-Holger Lange, Peter Ulbrich
Akademie der Wissenschaften der DDR,
Zentralinstitut für Kybernetik und Informationsprozesse

Kassettenbereiche in den RAM gelesen werden und nach einem Kennzeichen für den Systemlader überprüft werden. Es sind dies bei

– Diskette: Spur 0, Sektoren 1 ... 4
– Kassette: Block 0 ... 3.

Bei älteren Versionen von SIOS (bis Version 0.7) und UDOS (Version 2.12) wird der Systemlader auf der Diskette auf Spur 1, Sektor 1 ...7 gesucht. Wenn ein Systemlader gefunden wird, wird die weitere Suche abgebrochen. War die Suche erfolglos, dann blinken die drei roten Lampen der Fehleranzeige auf der Tastatur rhythmisch, wodurch signalisiert wird, daß kein Betriebssystem verfügbar ist. Hier hilft nur Datenträgerwechsel und ein erneutes RESET.

1.4.2.
Systemlader

Die Systemlader sind schon in bestimmten Parametern betriebssystemspezifisch. Ihnen ist folgender Aufbau gemeinsam:

1. – 3. Byte: Kennwort 'SYL'
4. \ Byte: Versionsnummer
5. – 6. Byte: Anzahl der zu ladenden Bytes (L-Teil, H-Teil)
ab 7. Byte: Laderoutine.

Die Laderoutine ist im allgemeinen in der Lage, bereits Dateien zu laden. Sie sucht die zu ladenden Komponenten auf der Diskette oder Kassette und transportiert sie an die dafür bestimmten Bereiche im Arbeitsspeicher. Auch das ist bei jedem der drei Betriebssysteme unterschiedlich. Anschließend werden in jedem Betriebssystem die spezifischen Betriebsbeginnroutinen angesprungen. Damit ist der Ladevorgang beendet, und alle weiteren Maßnahmen der Initialisierung laufen unter Regie des jeweiligen Betriebssystems ab.

2.
MUTOS 8000
2.1.
Übersicht

MUTOS 8000 ist das Betriebssystem des Bürocomputers A 5120.16. Grundlage des BC A 5120.16 ist ein BC A 5120 mit 64 K Byte RAM, großem Bildschirm (24 Zeilen mit je 80 Spalten), Tastatur K 7637, über serielles Interface angeschlossenem Drucker 1152 oder 1157, drei Standard-Floppy-Disk (zwei davon im Beistellgefäß) mit einfacher Schreibdichte. Die Aufwertung des BC A 5120 zum A 5120.16 ist durch ein am K-1520-Bus angeschlossenes Leiterkartenpaar gegeben, das einen 16-Bit-Mikroprozessor U 8001 und 256 K Byte RAM enthält. Der BC A 5120.16 kann als normaler BC mit einem der in diesem Heft vorgestellten Betriebssysteme SCP, UDOS oder SIOS betrieben werden. Dann werden der U 8001 und seine 256 K Byte RAM nicht genutzt, das Aufwertungskartenpaar wird nicht aktiv. Läuft der BC A 5120.16 jedoch unter MUTOS 8000, dann ist der U-8001-Prozessor der Haus-Prozessor des Betriebssystems, der 256 K Byte RAM ist der Arbeitsspeicher des Betriebssystems und der U-880-Prozessor des Bürocomputers dient als Ein/Ausgabe-Prozessor, der unter einem speziellen Steuerprogramm die gesamte Peripherie des Bürocomputers bedient. Der BC A 5120.16 ist also sowohl als 8-Bit-Mikrorechner, als auch als 16-Bit-Mikrorechner betreibbar. Auf Möglichkeiten, die 256 K Byte RAM und die Leistungen des U 8001 auch unter einem der 8-Bit-Betriebssysteme zugänglich zu machen, soll in diesem, dem MUTOS 8000 gewidmeten Artikel nicht eingegangen werden.

MUTOS 8000 ist ein universelles, interaktives Time-Sharing-System für den Ein- und Mehrnutzerbetrieb, das durch folgende Eigenschaften charakterisiert werden kann:

- Es besitzt ein hierarchisches Filesystem
- es hat eine einheitliche Ein/Ausgabe-Schnittstelle für Files, Geräte und Interprozeß-Kommunikation
- es bietet die Möglichkeit, asynchrone Prozesse zu initiieren und in existierenden Prozessen das laufende Programm auszutauschen.

MUTOS 8000 ist entstanden durch die Übertragung des Betriebssystems MUTOS 1630, das für den Basisrechner A 6402 (K 1630) des VEB Kombinat Robotron entwickelt wurde. MUTOS-Betriebssysteme sind kompatibel zum System INMOS für SKR-Kleinrechner und zum Betriebssystem UNIX[1], Version 7, das unter diesem oder unter anderen Namen auf praktisch allen Rechnern (von 16-Bit-Mikrorechnern an aufwärts bis zu Großrechnern) angeboten wird. Die Kompatibilität erstreckt sich sowohl auf die Schnittstelle Nutzer – Rechner (Kommandosprache, Kommandosatz, letzterer evtl. eingeschränkt), als auch auf die Schnittstelle Programm – Betriebssystem. MUTOS ist ferner in fast allen Punkten kompatibel mit dem System PSU (Programmierstützende Umgebung) des VEB Leitzentrum für Anwendungsforschung Berlin (LfA), das auf ESER-Anlagen unter den Betriebssystemen SVM-PTS, OS/ES-TSO bzw. im Stapelbetrieb unter OS/ES eingesetzt werden kann.

MUTOS ist ein angenehmes und leicht beherrschbares Time-Sharing-System für den professionellen Programmierer, der kooperativ in einer Gruppe arbeiten will. Seine primären Einsatzgebiete sind Programmentwicklung und Textverarbeitung bis zum Typesetting. Ein Parser-Generator, ein Generator für lexikalische Analysen, Makrogeneratoren und Werkzeuge für die Entwicklung großer Softwarepakete stehen zur Verfügung. Neben C werden zu einem späteren Zeitpunkt weitere höhere Programmiersprachen (PASCAL, PLZ/SYS, MODULA2, BASIC-Interpreter, FORTRAN 77, COBOL) bereitgestellt. Ein komfortables, mächtiges, relationales Datenbanksystem kann nachgenutzt werden.

Der MUTOS-Betriebssystem-Kern mit seinen ungefähr 10 000 Zeilen Quellcode, davon über 90 Prozent in C geschrieben, ist ein ideales Objekt für die

1 UNIX ist ein Warenzeichen der Bell Laboratories.

Informatik-Ausbildung, da er einerseits von einem einzelnen noch analysiert und verstanden werden kann, und weil MUTOS andererseits kein „Spiel-System" ist.
Da das zu MUTOS kompatible UNIX zu einem Quasi-Standard-Betriebssystem geworden ist, genießt der MUTOS-Nutzer alle Vorzüge eines Standards: Er findet überall die gleiche Umgebung vor und seine Programme laufen, wenn sie nur in C oder einer anderen höheren Programmiersprache entwickelt wurden, unabhängig von den Rechnern, auf denen sie ursprünglich entwickelt wurden.
MUTOS, seine Bibliotheken und seine Kommandos sind bis auf wenige Ausnahmen in der rechnernahen höheren Programmiersprache C geschrieben, so daß für die Übertragung von MUTOS 1630 auf den BC A 5120.16 neben den durch die unterschiedlichen Rechnerarchitekturen bedingten Anpassungsarbeiten im wesentlichen die Schaffung eines C-Compilers, eines PLZ/ASM-Assemblers und eines Verbinders (Linkage Editors) für den U 8000 erforderlich waren. Diese Komponenten wurden zuerst unter MUTOS 1630 als Cross-Varianten in C geschrieben und später zusammen mit dem mit ihrer Hilfe hergestellten MUTOS-8000-Betriebssystemkern auf den BC A 5120.16 übernommen. Wenigstens hier seien die nicht unerheblichen Arbeiten zur Schaffung des oben erwähnten Steuerprogramms für den 8-Bit-E/A-Prozessor (d. h. für den eigentlichen BC A 5120) erwähnt.
Der Name MUTOS ist aus *Multi User Time-Sharing Operating System* abgeleitet. MUTOS 1630 ist wie seine UNIX-Verwandtschaft ein auf schnelle Magnetplatten mit ausreichender Kapazität, zum Beispiel Winchester-Laufwerke, orientiertes Betriebssystem. Fehlen diese, ist der volle Leistungsumfang nicht – oder nur mit Unbequemlichkeiten verbunden – nutzbar. Da beim Transport von MUTOS 1630 auf den nur mit drei Floppy-Laufwerken ausgerüsteten BC A 5120.16 keines der grundlegenden MUTOS-Konzepte aufgegeben wurde, trifft obige Einschränkung auf MUTOS 8000 zu.
Zur Normalausstattung des BC A 5120.16 gehört kein zweites Terminal. Ist jedoch, wie bei den Autoren, ein zweites Terminal angeschlossen, gestattet MUTOS 8000 bei einigen Aufgabenklassen einen sinnvollen Mehrnutzer-Betrieb, zum Beispiel Editier-Arbeiten durch zwei Nutzer bei gleichzeitiger Textverarbeitung (runoff) mit Drucker-Ausgabe im Hintergrund.
Ein Vergleich von MUTOS 8000 mit den drei anderen in diesem Heft vorgestellten Betriebssystemen ist problematisch. Einerseits läuft MUTOS 8000 im Gegensatz zu jenen auf einem 16-Bit-Prozessor mit 4 MHz Taktfrequenz und 256 K Byte RAM, andererseits hätte eine Anpassung von MUTOS an die Peripherie-Bedingungen (kein großer schneller Sekundärspeicher) des BC A 5120.16 bedeutet, auf wesentliche Eigenschaften und Leistungen von MUTOS zu verzichten, so daß die speziell für den BC A 5120 entwickelten Betriebssysteme zumindest in einigen Punkten bequemer handhabbar sind. MUTOS 8000 auf dem BC A 5120.16 ist als eine Zwischenlösung zu betrachten, die **das** Standard-Betriebssystem für leistungsfähige 16-Bit-Rechner (und größere) schon vor der Bereitstellung von 16-Bit-Mikrorechnern mit komfortabler Plattenperipherie bei den Nutzern von Mikrorechnern einführt. Die Aufwertung eines der drei Systeme SCP, UDOS oder SIOS oder die Schaffung eines völlig neuen Betriebssystems für den BC A 5120.16 wäre in Anbetracht des dafür nötigen Aufwandes und der Stellung, die der BC A 5120.16 zwischen den 8-Bit-Rechnern und den speziell entwickelten 16-Bit-Mikrorechnern einnimmt, nicht zu rechtfertigen gewesen.
Die Problematik des Betriebssystemvergleichs kommt auch bei der vorgegebenen Gliederung zum Tragen, die den 8-Bit-Betriebssystemen Rechnung trägt. UNIX-kompatible Systeme, wie zum Beispiel MUTOS, brauchen keinen Vergleich mit beliebigen Time-Sharing-Betriebssystemen zu scheuen, im Gegenteil, in neueren Betriebssystem-Entwicklungen ist oft der UNIX-Einfluß spürbar. Für die Darstellung eines solchen Systems sind angepaßtere Gliederungen denkbar. Um nicht gänzlich am Leser vorbeizureden, werden im folgenden Überblick auch Eigenschaften dargestellt, die unter anderen Hauptgliederungspunkten erwartet werden können.

2.1.1. Nutzung des U-8001-Prozessors unter MUTOS 8000

Der U 8001 kann in verschiedenen Modi arbeiten. Im System-Modus sind alle Befehle zulässig, im Nutzer-Modus nur eine Teilmenge. Bei synchronen und asynchronen Unterbrechungen geht der Prozessor in den System-Modus über. Ist der Prozessor im Nutzer-Modus, führt die Abarbeitung privilegierter Befehle (z. B. E/A-Befehle oder Befehle für den Wechsel des Prozessor-Modus) zu einer Unterbrechung. Eine spezielle Operation, der *System-CALL*, dient dem gezielten Übergang aus dem Nutzer- in den System-Modus.
Unter MUTOS 8000 arbeiten alle Programme und Kommandos, das Kommando-Interpreter-Programm *Shell* eingeschlossen, im Nutzer-Modus. Jede Art von Diensten müssen diese Programme mit Hilfe von System-CALLs vom MUTOS-8000-Betriebssystemkern, der im System-Modus läuft, erbitten. Damit ist auch schon gesagt, daß der MUTOS-8000-Kern für die Behandlung aller Unterbrechungen zuständig ist.
Der U-8001-Prozessor kennt zwei weitere Modi. Im *segmentierten* Modus ist einem Programm der gesamte angeschlossene Hauptspeicher zugänglich. Im *nichtsegmentierten* Modus steht einem Programm nur ein Adreßraum von 64 K Byte zur Verfügung. Unter MUTOS 8000 wird der Prozessor, von ganz wenigen Ausnahmen abgesehen, nur im *nichtsegmentierten* Modus betrieben.
Durch die Hardware bedingt ist der 256-KByte-RAM in vier Segmente zu je 64 K Byte aufgeteilt, von denen eines dem Betriebssystemkern vorbehalten ist. In den drei verbleibenden Segmenten kann jeweils ein Nutzerprogramm untergebracht werden. Ein Nutzerprogramm kann deshalb weder den Betriebssystem-Kern noch andere Nutzer-

programme durch Lese- oder Schreibbefehle erreichen.

2.1.2.
Das Filesystem

Vom Nutzerstandpunkt aus gesehen gibt es drei Arten von Files:
gewöhnliche Plattenfiles, Verzeichnisse (directories) und spezielle Files. Innerhalb eines Filesystems ist jedes File durch eine Zahl, die Index-Nummer, eindeutig identifiziert.

Gewöhnliche Files

Ein File enthält das, was der Nutzer, oder genauer, ein Programm in ihm ablegt, zum Beispiel den Text eines Artikels oder ein ausführbares Programm. Ein Text-File besteht aus einer Zeichenkette, Zeilen werden durch das ASCII-Zeichen line-feed begrenzt.
Ein ausführbares Programm besteht aus Binärdaten, wie sie bei der Programmausführung im Hauptspeicher erscheinen werden. Einige Programme benötigen stärker strukturierte Files: Der Assembler erzeugt zum Beispiel ein Objektfile mit einer solchen Struktur, wie sie der Verbinder erwartet. Es gilt jedoch: Die Struktur gewöhnlicher Files wird von den Programmen festgelegt, die sie erzeugen, nicht vom Betriebssystem. Für das Betriebssystem ist jedes gewöhnliche File eine Bytefolge bestimmter Länge, eventuell mit Lücken. Die Länge eines gewöhnlichen Files kann sich während seiner Lebenszeit ändern.

Verzeichnisse

Verzeichnisse sorgen für eine Verknüpfung von File-Namen mit den Files selbst, das heißt, sie bilden Namen auf Index-Nummern ab. Durch Verzeichnisse wird das File-System strukturiert.
Jeder Nutzer hat ein Verzeichnis für seine eigenen Files. Er kann Unterverzeichnisse erzeugen für Gruppen von Files, die er als eine Einheit betrachten möchte. Ein Verzeichnis verhält sich genauso wie ein gewöhnliches File, mit einer Ausnahme: Nur der Betriebssystem-Kern darf auf ein Verzeichnis-File schreiben. Jedoch darf jedes Programm mit der nötigen Erlaubnis ein Verzeichnis wie ein gewöhnliches File lesen.
Das Betriebssystem unterhält einige Verzeichnisse für den eigenen Bedarf. Das wichtigste ist das Wurzel-Verzeichnis (root directory). Alle Files im File-System können gefunden werden, indem ein Weg durch eine Kette von Verzeichnissen verfolgt wird, bis das gewünschte File erreicht ist. Der Ausgangspunkt für eine solche Suche ist oft das Wurzel-Verzeichnis.
Andere Verzeichnisse enthalten die Namen von häufig benutzten Programmen, das heißt die Namen von Kommandos. Zur Ausführung eines Programms ist es jedoch nicht nötig, daß sein Name in einem dieser Verzeichnisse vermerkt ist.
File-Namen bestehen aus ein bis maximal 14 Zeichen. Der Name eines Files kann dem System in Form eines Pfad-Namens mitgeteilt werden. Ein Pfad-Name ist eine Folge von Verzeichnis-Namen, die durch "/" getrennt sind, gefolgt von einem File-Namen. Beginnt der Pfad-Name mit "/", so beginnt der Pfad im Wurzel-Verzeichnis. Der Name **/Maier/mueller/abc** veranlaßt das System, im Wurzel-Verzeichnis das Verzeichnis **Maier**, im Verzeichnis **Maier** das Verzeichnis **mueller** und im Verzeichnis **mueller** das File **abc** zu suchen. **abc** kann nun gewöhnliches File, Verzeichnis oder spezielles File sein. Der Name "/" bezeichnet das Wurzel-Verzeichnis.
Ein Pfad-Name, der nicht mit "/" beginnt, veranlaßt das System, seine Suche im aktuellen Verzeichnis des Nutzers zu beginnen. Zum Beispiel führt **a/b** über das Unterverzeichnis **a** des laufenden Verzeichnisses zu File **b**. Die einfachste Form eines Namens, zum Beispiel **alpha**, bezeichnet ein File, das im laufenden Verzeichnis gefunden wird. Ein Null-Name (Länge 0) bezeichnet das laufende Verzeichnis.
Alle gewöhnlichen und speziellen Files können in verschiedenen Verzeichnissen unter möglicherweise verschiedenen Namen erscheinen. MUTOS unterscheidet sich von anderen Betriebssystemen darin, daß alle Verweise (links) auf ein File gleichberechtigt sind. Daraus folgt, daß ein File nicht innerhalb eines Verzeichnisses existiert, sondern eine vom Namen unabhängige Größe ist. Eine Verzeichnis-Eintragung besteht auch tatsächlich nur aus einem 14-Byte-Namensfeld und einer 2-Byte-File-Index-Nummer. Allerdings endet die Existenz eines Files, sobald der letzte Verzeichnis-Verweis auf das File gelöscht ist und kein Prozeß mehr mit dem File arbeitet.
Jedes Verzeichnis hat mindestens zwei Eintragungen: „." und „..".
„." ist Synonym für das Verzeichnis selbst, „.." ist Synonym für das übergeordnete Verzeichnis. (Nur im Wurzel-Verzeichnis bezeichnet auch „.." das Wurzel-Verzeichnis selbst.)
Die Verzeichnisse bilden einen Wurzelbaum. Abgesehen von den speziellen Eintragungen „." und „.." muß jedes Verzeichnis in genau einem Verzeichnis, seinem übergeordneten Verzeichnis, eingetragen sein.
Zur Erläuterung dieses Abschnittes folgen einige Beispiele für die Anwendung der Kommandos **cd** (change current directory), **ls** (list directory) und **pwd** (print working (=current) directory):

pwd – Der vollständige Pfad-Name des laufenden Verzeichnisses wird ausgegeben. Er sei **/usr/bummi/test**.
ls – Alle im laufenden Verzeichnis (**/usr/bummi/test**) enthaltenen File-Namen werden ausgegeben
ls . – Erzeugt die gleiche Liste wie oben.
ls / – Erzeugt die Liste der im Wurzel-Verzeichnis eingetragenen Namen.
ls .. – Die Liste aller File-Namen im Verzeichnis **/usr/bumml** wird ausgegeben.
cd .. – **/usr/bumml** wird das laufende Verzeichnis.
pwd – Liefert jetzt **/usr/bummi.**
ls bzw. **ls .** bzw. **ls /usr/bumml** – Die gleiche Liste wird dreimal erzeugt.
ls test bzw. **ls ./test** – Die im Unterverzeichnis **test** enthaltenen File-Namen werden zweimal ausgegeben.
cd ../.. – Das Wurzel-Verzeichnis wird laufendes Verzeichnis.
pwd – liefert „/"
cd /usr/bummi/test1 – Wir sind in einem anderen Unterverzeichnis von **/usr/bumml.**
ls ../../../usr/bumml/test. – Ein höchst umständlicher Weg, das Verzeichnis **test** aufzulisten.
ls ../test – hätte es auch getan.

Spezielles Files

Zu jedem vom System unterstützten Ein-/Ausgabegerät gibt es mindestens ein spezielles File. Spezielle Files wer-

den gelesen oder beschrieben wie gewöhnliche Files, aber Lese-/Schreibanforderungen bewirken die Aktivierung des jeweiligen Gerätes. Eine Eintragung für jedes spezielle File existiert im Unterverzeichnis **dev** des Wurzel-Verzeichnisses. Jedoch können auch alle anderen Verzeichnisse Verweise (links) auf spezielle Files enthalten. Um zum Beispiel auf den Drucker auszugeben, würde man auf das File **/dev/lp** schreiben.

Spezielle Files existieren zum Beispiel für jedes Terminal, jede Floppy-Einheit und für den physischen Hauptspeicher. Natürlich werden aktive Floppy-Einheiten und der Hauptspeicher vor unzulässigen Zugriffen geschützt.

Ein-/Ausgabegeräte auf diese Weise zu behandeln hat drei Vorzüge:

- File- und Geräte-Ein/Ausgabe sind so ähnlich wie möglich.
- File- und Geräte-Namen haben die gleiche Syntax und Semantik, so daß einem Programm, das einen File-Namen als Parameter erwartet, ein Geräte-Name übergeben werden kann.
- Spezielle Files unterliegen demselben Schutzmechanismus wie gewöhnliche Files.

Montierbare File-Systeme

Das Wurzel-Verzeichnis liegt immer auf dem gleichen Gerät. Aber nicht die gesamte File-System-Hierarchie muß auf einem Gerät residieren. Es gibt einen **mount**-Systemruf mit zwei Argumenten:

- dem Namen eines existierenden gewöhnlichen Files oder Verzeichnis-Files
- dem Namen eines speziellen Files, auf dessen zugeordnetem Gerät (z. B. ein Plattenlaufwerk) ein Datenträger mit einem unabhängigen File-System (mit eigener Verzeichnis-Hierarchie) liegt.

Der Effekt des **mount**-Rufs besteht darin, ein Blatt (oder einen Knoten im Fall eines Verzeichnis-Namens) des File-System-Baums durch einen ganzen Unterbaum zu ersetzen. Das wird erreicht, indem das System jeden Verweis auf das im **mount**-Ruf spezifizierte File in einen Verweis auf das Wurzel-Verzeichnis des montierten File-Systems umwandelt.

Ein montierbares File-System kann durch geeignete Schreiboperationen auf ein spezielles File erzeugt werden, zum Beispiel mit dem Programm **mkfs.**

Nach einem **mount** gibt es, vom Nutzerstandpunkt aus gesehen, keinen Unterschied mehr zwischen Files des permanenten File-Systems und Files des montierbaren File-Systems. Sie alle sind in einer Hierarchie vereinigt. (Einzig Verweise (links) dürfen die Grenzen zwischen den ehemals getrennten File-Systemen nicht überschreiten. Dafür sorgt aber das System selbst, weil nur so Schwierigkeiten bei späteren **umount**-Aktionen umgangen werden können.)

Das Schutzsystem

Jeder Nutzer des Systems hat eine eindeutige Nutzer-Identifikation (Nutzer-ID). Ebenso gehört jeder Nutzer zu einer Nutzer-Gruppe mit zugehöriger Gruppen-ID.

Wird ein File erzeugt, werden Nutzer-ID und Gruppen-ID des File-Besitzers in die File-Beschreibung übernommen. Außerdem werden zehn Schutzbits Teil der File-Beschreibung. Die Belegung der Schutzbits ist jedem Besitzer eines Files überlassen. Neun dieser Bits spezifizieren unabhängig voneinander die Lese-, Schreib- und Ausführberechtigung für den File-Besitzer, für Mitglieder seiner Gruppe und für die Allgemeinheit. Bei Verzeichnis-Files bedeutet Ausführberechtigung die Berechtigung, einen Pfad-Namen über dieses Verzeichnis zu führen. Für ausführbare Programme ist es die Berechtigung, sie auszuführen.

Das zehnte Schutzbit ist nur für die ausführbaren Files von Bedeutung. Ist dieses Bit eingeschaltet und wird das File (Programm) vom System ausgeführt, so ersetzt das System während der Laufzeit des Programms den systemintern geführten aktuellen Nutzer-ID durch den Nutzer-ID des Besitzers des Files (Programms).

Das System kennt einen speziellen Nutzer-ID, den des *super-user,* für den die üblichen Beschränkungen beim File-Zugriff nicht gelten. Ebenso werden gewisse Systemdienste nur dem *super-user* gewährt. Durch den mit dem zehnten Schutzbit gebotenen Dienst ist es möglich, sorgfältig geschriebene Programme, die privilegierte Systemrufe enthalten, allen Nutzern zur Verfügung zu stellen. Voraussetzung ist, daß der Besitzer des Programms, der *superuser,* das zehnte Schutzbit eingeschaltet hat. Weitere Anwendungen für dieses *Set-User-ID-Bit* sind in Verbindung mit den Zugriffsrechten für Files gegeben.

Ein-/Ausgabe-Systemrufe

Ein-/Ausgabe-Rufe sind unter MUTOS wie alle anderen Systemrufe analog zu normalen Funktionsrufen gestaltet:

function-value
= function (argument...).

Es gibt keinerlei Steuerblöcke im Nutzerprogrammbereich (im Adreßraum des Nutzerprogramms).

Jeder Systemruf kann als Wert auch einen Fehlerkode haben, in der Regel eine negative Zahl. Im folgenden wird diese Möglichkeit nicht weiter verfolgt.

Die Ein-/Ausgabe-Rufe sind so konzipiert, daß keine Unterschiede zwischen verschiedenen Gerätetypen und Zugriffsformen gemacht werden müssen. Damit gibt es keinen Unterschied zwischen sequentiellem und direktem Zugriff, noch gibt es irgendeine Form logischer Satzlängen. Die Größe eines Files ist bestimmt durch die Zahl der Bytes, die bereits auf das File geschrieben wurden.

Soll ein existierendes File gelesen oder soll auf ein solches File geschrieben werden, muß es durch folgenden Ruf eröffnet werden:

filed = open (filename, flags).

Dabei ist filename der Name des Files im laufenden Verzeichnis oder ein beliebiger Pfad-Name, der zu dem File führt. Das flag-Argument zeigt an, ob Lese- und/oder Schreibzugriffe auf das File geplant sind. Der Funktionswert filed (file descriptor) ist eine kleine ganze Zahl, die das Programm im weiteren zur Identifikation des Files bei Lese- und Schreibzugriffen oder bei anderen System-Rufen benötigt.

Unter MUTOS kann wie in den meisten UNIX-Systemen ein Programm bis zu zwanzig Files gleichzeitig eröffnet haben, das heißt, die Deskriptoren sind Zahlen zwischen 0 und 19. Der kleinste freie Deskriptor wird bei einem open-Ruf als Wert zurückgegeben.

Der create-Systemruf kann benutzt werden, um ein neues File zu erzeugen bzw. ein existierendes File auf die Länge Null zu verkürzen. Create eröffnet gleichzeitig das neue File für Schreiboperationen und liefert als Funktionswert, ebenso wie open, einen File-Deskriptor. Argument des create-Rufes ist wieder ein File-Name, der auch ein Pfad-Name sein kann.

Es gibt keinerlei Beschränkung für die Zahl der Programme, die gleichzeitig ein File für Lese- oder Schreiboperationen eröffnet haben können. Die systeminternen Schutzmechanismen garantieren die logische Konsistenz des File-Systems, wenn zwei Programme gleichzeitig auf ein File schreiben oder Files im selben Verzeichnis erzeugen oder einander die eröffneten Files entfernen. Darüber hinausgehender Schutz ist unter den typischen MUTOS-Einsatzbedingungen selten erforderlich und mit einfachen Mitteln nicht erreichbar.

Lesen und Schreiben erfolgt sequentiell in den vom Programm gewünschten Einheiten. Das heißt, wenn ein Byte im File das letzte Byte im File war, das mit einem Ein-/Ausgabe-Ruf gelesen oder geschrieben worden ist, dann bezieht sich der nächste Ein-/Ausgabe-Ruf implizit auf das diesem Byte unmittelbar folgende Byte. Für jedes von einem Programm eröffnete File weist ein systemintern geführter Zeiger auf das nächste zu lesende oder zu schreibende Byte. Werden n Bytes gelesen oder geschrieben, so rückt der Zeiger um n Bytes vor. Auf ein eröffnetes File kann mit folgenden Rufen zugegriffen werden:

n = read (filed, buffer, m)
n = write (filed, buffer, m).

Bis zu m Bytes werden zwischen dem über filed spezifizierten File und dem über buffer spezifizierten Ein-/Ausgabebereich des Programms übertragen. Der Wert n des Rufs ist die Zahl der tatsächlich übertragenen Bytes. Im Fall einer Schreiboperation ist n gleich m, es sei denn, eine Ausnahmebedingung – zum Beispiel Ein-/Ausgabefehler oder Ende des physikalischen Mediums bei speziellen Files – ist eingetreten.

Bei einer Leseoperation kann n kleiner als m sein, ohne daß ein Fehler aufgetreten ist. Wenn der systeminterne Zeiger dem File-Ende so nahe ist, daß nicht mehr m Bytes gelesen werden können, ohne das File-Ende zu überschreiten, dann werden nur noch die verbleibenden Bytes des Files übertragen. Fernschreiberähnliche Terminals liefern auf eine Leseanforderung niemals mehr als eine Eingabe-Zeile. Wenn ein Lese-Ruf einen Wert n gleich Null liefert, ist entweder der interne Zeiger gleich der aktuellen File-Größe, das heißt, End of File ist erreicht, oder eine Fluchtsequenz wurde am Terminal eingegeben, um im Programm Fileende anzuzeigen.

Geschriebene Bytes verändern nur den Teil eines Files, auf den der interne Zeiger vor einem Schreib-Ruf zeigt. Weist der Zeiger hinter das letzte Byte des Files, wird das File entsprechend erweitert. Um einen Direktzugriff auf eine File zu erreichen, muß vor einem Lese- oder Schreib-Ruf der systeminterne Zeiger auf die gewünschte Stelle im File gerichtet werden:

location = lseek (filed, offset, base).

Abhängig vom Wert von base wird der interne Zeiger auf eine Position im File gerichtet, die um offset Bytes vom Beginn oder vom Ende des Files oder von der gerade eingenommenen Position entfernt ist. Offset darf negativ sein. Location ist die Position, vom File-Beginn an gerechnet, auf die der Zeiger nach Ausführung des lseek-Rufes weist. Für gewisse spezielle Files, zum Beispiel Terminals, wird der lseek-Ruf vom System ignoriert.

Der lseek-Ruf kann benutzt werden, um gewöhnliche Files mit Lücken zu erzeugen. Lücken belegen keinen Platz im File-System. Sie werden als binäre Nullen gelesen.

Weitere Systemrufe, die hier nicht weiter betrachtet werden sollen, haben mit Files zu tun, zum Beispiel:

– Schließen eines Files
– den Status eines Files anzeigen
– den Besitzer eines Files oder die Schutzbits eines Files modifizieren
– in einem Verzeichnis einen Verweis (link) zu einem existierenden File einrichten oder löschen.

Die Implementierung des File-Systems

Eine Platte oder Diskette wird in eine geräteabhängige Zahl von Blöcken je 512 Byte zerlegt. Die Blöcke werden über logische Blocknummern 0, 1, 2 usw. adressiert. Block 0 wird vom File-system nicht belegt. Block 1, der *superblock*, enthält Angaben zur Größe des File-Systems und zur Zahl der Index-Blöcke, den Anker der Freispeicher-Liste sowie weitere Informationen, die hier nicht interessieren sollen.

Die Index-Blöcke (Blocknummern 2, 3 usw.) enthalten die schon im Abschnitt *Schutzsystem* eingeführten File-Beschreibungen. Eine File-Beschreibung (inode – index node) ist 64 Bytes lang. Wie bereits erwähnt, ist jedes File durch eine Index-Nummer eindeutig identifiziert. Tatsächlich befinden sich die File-Beschreibungen der Files mit den Index-Nummern 1 bis 8 im Block 2, die inodes zu den Index-Nummern 9 bis 16 im Block 3 usw. Die Zahl der Index-Blöcke wird bei der Initialisierung eines File-Systems festgelegt. Der übliche Ansatz ist, soviel Index-Blöcke bereitzustellen, daß bei völlig ausgeschöpftem File-System ein File im Mittel 1536 Bytes lang sein kann. Eine File-Beschreibung (inode) enthält folgende Daten:

– den Nutzer- und den Gruppen-ID des File-Besitzers
– die Schutzbits
– die Nummern der Blöcke, in denen die Bytes des Files abgelegt sind
– die Zeitpunkte der File-Erzeugung, des letzten Zugriffs auf das File und der letzten Modifikation des Files
– die Zahl der Verweise (links) auf das File, das heißt eine Angabe, aus wieviel Verzeichnissen auf das File verwiesen wird
– Angaben zur Art des Files, ob es sich um ein gewöhnliches File, ein Verzeichnis-File oder ein spezielles File handelt.

Durch einen open- oder create-Systemruf wird die einem Pfad-Namen zugeordnete Index-Nummer bestimmt, indem die explizit oder implizit angegebenen Verzeichnisse durchsucht werden.

Die zur Index-Nummer gehörende File-Beschreibung wird vom System eingelesen und in einer internen Tabelle abgelegt.

Bis zum close-Zeitpunkt, eventuell auch länger, enthält dann die hauptspeicherresidente Filebeschreibung die gültige Beschreibung des Files.
In einer weiteren internen Tabelle wird für jeden open- oder create-Ruf
– die Index-Nummer der nun residenten Filebeschreibung
– das Gerät, auf dem das File residiert und
– der Lese-/Schreib-Zugriffszeiger
abgelegt. Der beim Eröffnen zurückgegebene File-Deskriptor gestattet dem System den raschen Zugriff auf die zum File und zum eröffnenden Prozeß gehörende Eintragung in dieser Tabelle. (Der File-Deskriptor ist Index für eine prozeßbezogene Tabelle, die nur Verweise auf solche Eintragungen enthält.)
Wird ein File erzeugt (create), dann wird eine freie File-Beschreibung (inode) bestimmt und in dem Verzeichnis, das laut Pfad-Name dafür vorgesehen ist, eine Eintragung vorgenommen, die, wie schon erwähnt, Index-Nummer und File-Namen enthält. Ein Verweis (link) auf ein existierendes File wird eingerichtet, indem im angegebenen Verzeichnis eine Eintragung erfolgt, die die Index-Nummer des Files enthält, und indem in der File-Beschreibung der Verweis-Zähler um Eins erhöht wird.
Ein File wird entfernt, indem der Verweis-Zähler in der File-Beschreibung um Eins verringert wird und die entsprechende Verzeichnis-Eintragung gelöscht wird. (Genauer, das Index-Nummernfeld der Eintragung wird auf Null gesetzt.)
Wird der Verweis-Zähler Null, so werden die dem File zugeordneten Datenblöcke und die File-Beschreibung freigegeben.
Eine File-Beschreibung enthält 13 Felder zu 3 Bytes, die Blocknummern aufnehmen können. Für gewöhnliche Files und Verzeichnisse gilt:

- Die ersten zehn Felder enthalten die Nummern der ersten zehn Datenblöcke des Files. (Die Zahl Null zeigt dabei an, daß kein Datenblock vorhanden ist.) Damit kann das System auf die ersten 5 K Byte eines Files direkt zugreifen, sobald die File-Beschreibung im Hauptspeicher ist.
- Das elfte Feld enthält die Nummer eines Indirekt-Blockes, das heißt die Nummer eines Blockes, der die Blocknummern der nächsten 128 Datenblöcke des Files enthält.
- Das zwölfte Feld enthält die Nummer eines doppelten, das dreizehnte Feld die Nummer eines dreifachen Indirekt-Blockes. Damit können mit einer File-Beschreibung Files bis zu einer Länge von $512 * (10 + 128 + 128^2 + 128^3)$ Bytes beschrieben werden. Bis zu vier Zugriffe werden also benötigt, um auf ein Byte eines Files zugreifen zu können. Diese Zahl wird jedoch oft reduziert durch einen internen Puffer-Mechanismus, der häufig benutzte Blöcke im Hauptspeicher hält.

Die File-Beschreibungen spezieller Files enthalten nur im ersten der 13 Felder eine relevante Information, nämlich einen internen Gerätenamen des dem speziellen File zugeordneten Geräts. Der Gerätename besteht aus zwei Zahlen, die den Gerätetyp und die Geräteeinheitnummer bezeichnen.
In einer internen Tabelle werden zu jedem Gerätetyp die Adresse der für open-, read-, write-Rufe usw. zuständigen Routinen gehalten.
Für Nutzerprogramme scheint das Ein-/Ausgabesystem von MUTOS ungepuffert und synchron zu arbeiten. Das heißt, das System stellt nur soviel Bytes bereit, wie angefordert werden, und das Programm bekommt bei einem Systemruf erst dann die Steuerung zurück, wenn die Operation beendet ist. Intern verfolgt das System jedoch eine read-ahead und write-behind Strategie.

2.1.3. Prozesse unter MUTOS

Unter MUTOS arbeiten stets mehrere Prozesse (tasks) parallel. Natürlich hat zu jedem Zeitpunkt jeweils nur ein Prozeß die Steuerung. Jeder Prozeß kann sich im Nutzer- oder im System-Modus befinden. Der Übergang vom Nutzer- in den System-Modus erfolgt synchron über Traps bzw. asynchron durch Interrupts. Ein Prozeß wird charakterisiert durch seine Umgebung:
– die HS-Belegung des Prozesses
– die Inhalte der allgemeinen Register
– den Status des eröffneten Files
– das aktuelle (laufende) Verzeichnis
– den Nutzer-ID des Prozeß-Besitzers
usw.
Während der Code eines Prozesses, der im Nutzer-Modus arbeitet, prozeßabhängig ist, teilen alle Prozesse den gleichen Code (der zum Beispiel alle Geräte-Treiber enthält), wenn sie im System-Modus arbeiten. Die Systemteile der Prozesse kommunizieren miteinander durch gemeinsam genutzte (globale) Daten, wie zum Beispiel die inode-Tabelle, den E/A-Pufferbereich usw. Jeder Prozeß im System-Modus hat seinen eigenen System-Stack, der Teil der Prozeßumgebung ist. (Es gibt keinen prozeßunabhängigen Stack.) Wenn der Prozessor im Auftrag eines Prozesses arbeitet, muß die Prozeßumgebung im Hauptspeicher residieren. Wenn andere Prozesse aktiv sind, verbleibt die Umgebung gewöhnlich im HS, es sei denn, ein abarbeitungsbereiter Prozeß höherer Priorität bewirkt, daß die Prozeßumgebung ausgelagert wird (swapping).

Erzeugung neuer Prozesse

Nach dem Hochlaufen des Systems entstehen neue Prozesse nur durch Geben des fork-Systemrufs:
processid = fork ( ).
Bei Ausführung von fork zerfällt der aktive Prozeß in zwei Prozesse, die unabhängig voneinander um die Rechner-Ressourcen konkurrieren. Die Umgebungen der beiden Prozesse sind nahezu identisch, sie teilen zum Beispiel alle eröffneten Files. Der einzige Unterschied besteht darin, daß der eine als Eltern-Prozeß, der andere als Kind-Prozeß angesehen wird. Beide Prozesse führen ihre Arbeit mit den Instruktionen nach dem fork-Ruf durch. Und nur am Funktionswert des fork-Rufs, 0 im Kind-Prozeß, dem Prozeßidentifikator des Kind-Prozesses im Eltern-Prozeß, können die beiden Prozesse erkennen, wer welcher ist.

Kommunikation zwischen Prozessen – pipes

Ein Systemruf
fildesr, fildesw = pipe ( )
liefert zwei File-Deskriptoren, die anschließend für ganz normale read- bzw.

write-Rufe benutzt werden können. Ein read-Ruf in einem Prozeß mit dem File-Deskriptor **fildesr** wartet, bis in einem anderen Prozeß mit dem File-Deskriptor **fildesw** geschrieben wurde. Erst dann werden die Daten zwischen den beiden Prozessen bewegt. Prozesse, die auf diese Art miteinander kommunizieren wollen, müssen jedoch von einem gemeinsamen Vorgänger ins Leben gerufen worden sein, und in diesem Vorgänger muß schon der pipe-Ruf erfolgt sein.

Ausführung von Programmen

Das im Nutzer-Modus eines Prozesses laufende Programm kann mit einem execute-Ruf ausgetauscht werden. Gibt ein Programm den Ruf

execute (filename, arg1, arg2, ... argn),

so ersetzt das System das laufende Programm durch das mit filename benannte Programm und übergibt diesem die in Form von Zeichenketten spezifizierten Argumente arg1, arg2, ... argn. Die sonstige Prozeßumgebung bleibt beim execute-Ruf erhalten, das heißt insbesondere die eröffneten Files mit ihren File-Deskriptoren, das laufende Verzeichnis, der Prozeß-Besitzer, die inter-process-Beziehungen usw. Nur der eigentliche Programmtext und die Daten werden ausgetauscht. Das rufende Programm enthält nach dem execute-Ruf nur dann die Steuerung zurück, wenn zu filename kein File gefunden wurde oder wenn es für das File keine Ausführ-Berechtigung hatte.

Prozeßsynchronisation

Mit dem wait-Ruf

processid = wait (status)

kann ein Prozeß auf die Beendigung eines untergeordneten Prozesses warten. Der Funktionswert identifiziert den beendeten Kind-Prozeß, im Feld status wird ein Rückkehrcode des Kind-Prozesses abgelegt.

Beendigung eines Prozesses

Ein Prozeß beendet seine Existenz, wenn er im Nutzer-Modus den Ruf

exit (return-code)

gibt. Der Eltern-Prozeß wird darüber informiert, sobald er den wait-Ruf gibt. Alle eröffneten Files des beendeten Prozesses werden geschlossen und die Prozeß-Umgebung verschwindet. Prozesse können auch als Antwort auf illegale Aktionen oder durch Signale, die vom Terminal-Nutzer oder von anderen Prozessen erzeugt wurden, beendet werden.

2.2. System-Initialisierung

Da der Basis-Rechner des BC A 5120.16 ein BC A 5120 ist, erfolgt die Initialisierung von MUTOS 8000 unter einem Betriebssystem des BC A 5120, aus historischen Gründen unter UDOS.

Auf einer UDOS-Diskette befinden sich als Procedure-Dateien ein Ladeprogramm für den U-8000-RAM (LOAD 16), der MUTOS-8000-Betriebssystemkern (mutos), das Steuerprogramm für den U 880 (MUTOSEA), wenn dieser als E/A-Prozessor des U 8000 fungieren soll, Gerätetreiber, die der Treiberschnittstelle von UDOS genügen und eine DO-Datei (MUTOSBOOT), deren Kommandos den Anlauf von MUTOS 8000 einleiten. Unter laufendem UDOS wird mit DO MUTOSBOOT die Initialisierung von MUTOS 8000 gestartet. Das hat zur Folge, daß LOAD 16 „mutos" in das Segment 0 (Adressen 0 bis 64 K) des 256-KByte-RAM des U 8001 überträgt und Gerätetreiber sowie MUTOSEA in den 64-KByte-RAM des U 880 geladen werden. Anschließend wird der Nutzer über Bildschirmausschriften aufgefordert, sämtliche UDOS-Disketten aus den Disketten-Laufwerken zu entfernen und in Laufwerk Null das MUTOS-*root file system* einzulegen.

Ist der Diskettenaustausch erfolgt, kann der Nutzer durch Eingabe eines Leerzeichens MUTOSEA, das Steuerprogramm des MUTOS 8000 E/A-Prozessors, starten. Nach notwendigen Initialisierungsarbeiten erzeugt MUTOSEA mit einem Ausgabe-Befehl ein Reset für den U 8001 und startet damit den Betriebssystemkern von MUTOS 8000. Von diesem Zeitpunkt an realisiert MUTOSEA nur noch E/A-Anforderungen des MUTOS-Kerns bzw. leitet Tastatureingaben an den MUTOS-Kern weiter.

Es ist möglich, die soeben beschriebenen Aktivitäten bis zur Bildschirm-Ausgabe in die Initialisierungs-Sequenz von UDOS aufzunehmen, wenn die oben aufgeführten Dateien auf einer UDOS-Systemdiskette untergebracht werden. Dann wäre diese UDOS-MUTOS-Diskette in ein Laufwerk einzulegen, ein *Reset* für den BC A5120.16 zu geben und auf die über Bildschirm ausgegebene Aufforderung zu warten, die UDOS-Diskette zu entfernen.

Sobald der U-8001-Prozessor durch MUTOSEA aktiviert wurde, beginnt die Initialisierung des Kerns von MUTOS 8000. Globale Datenbereiche werden mit Startwerten versehen, der Superblock und die File-Beschreibung (inode) des Wurzelverzeichnisses (root directory) des MUTOS-*root file system* werden eingelesen. Ein Prozeß Null, der Scheduler-Prozeß, der nur im System-Modus und niemals im Nutzer-Modus läuft, wird eingerichtet. Bevor dieser Prozeß das eigentliche Scheduler-Programm betritt, das periodisch den Prozeß höchster Priorität aktiviert, erzeugt er noch über ein internes *fork* einen Kindprozess, der einen Systemruf execute */etc/init* absetzt, sobald er in den Nutzermodus gelangt. Das Programm *init*, das im Unterverzeichnis *etc* des Wurzelverzeichnisses eingetragen ist, hat die Aufgabe, für jedes angeschlossene Terminal einen Kindprozeß zu erzeugen. Welche Terminals angeschlossen sind, bei MUTOS 8000 wahrscheinlich nur eins, entnimmt */etc/init* dem File */etc/ttys*. In jedem der so erzeugten Kindprozesse wird mit *execute* das Programm */bin/login* gestartet. */bin/login* meldet sich am Terminal mit der Anschrift *login:* und der Nutzer muß seinen Namen und wenn er es so gewollt hat, ein Passwort eingeben. Im File */etc/passwd* sind für jeden zugelassenen Nutzer Name, Identifikatoren, der Name des zuständigen Kommando-Interpreters, der Name des Verzeichnisses, das bei Arbeitsbeginn aktuelles Verzeichnis sein soll und evtl. das verschlüsselte (!) Passwort eingetragen. *login* wertet die Tastatur-Eingabe aus. Ist sie korrekt, wechselt *login* mit einem *change current directory*-Systemruf das aktuelle Verzeichnis und aktiviert mit *execute* den dem Nutzer zugeordneten Kommando-Interpreter. Unter MUTOS 8000 wird der Kommando-Interpreter höchstwahrscheinlich *Shell*, das heißt

das unter */bin/sh* geführte Programm sein. Bevor Shell mit einem Prompt-Zeichen dem Nutzer anzeigt, daß er nun mit dem System kommunizieren kann, arbeitet Schell die Kommandos ab, die im laufenden Verzeichnis evtl. in einem File des Namens *Profile* gefunden werden. Damit ist die System-Initialisierung abgeschlossen.

Natürlich sind die Programme */etc/init*, */bin/login* und */bin/sh* nicht willkürlich durch andere Programme ersetzbar, aber es soll hier betont werden, daß all diese Programme im Nutzer-Modus arbeiten und wie jedes andere Programm nur auf die System-Rufe zurückgreifen können, wenn sie irgend etwas bewirken wollen.

Bei Abschluß der System-Initialisierung sind mindestens drei Prozesse aktiv:

- Prozeß Null, der Scheduler-Prozeß im System-Kern
- Prozeß Eins, der Prozeß, in dem */etc/init* läuft und auf die Beendigung eines seiner Kindprozesse wartet
- für jedes angeschlossene Terminal ein Prozeß, in dem entweder noch */bin/login* oder schon ein Kommando-Interpreter, meist */bin/sh* läuft.

Jeder aktive Prozess ist entweder ausgelagert oder belegt eines der vier Segmente des 256-KByte-RAM. Der Scheduler-Prozess wird niemals ausgelagert.

2.3.
Disketten und Dateien

Auf dem BC 5120.16 sind MUTOS-Filesysteme auf Disketten untergebracht. Der logische Aufbau eines MUTOS-Filesystems wurde schon unter Punkt 2.1.2. behandelt, dort aber eher aus allgemeiner MUTOS-Sicht. Hier sollen jetzt MUTOS-8000-Spezifika Aufnahme finden.

MUTOS 8000 nutzt Standard-Disketten (einfache Schreibdichte) mit 26 Sektoren zu je 128 Byte auf jeder der 77 Spuren. Spur Null wird nicht benutzt, um mit MUTOS 1630 kompatibel zu bleiben. Für MUTOS sind Blöcke á 512 Byte die kleinste Einheit auf einem externen Direktzugriffsspeicher. Das heißt, daß jeweils vier Sektoren einer MUTOS-Diskette einen Block bilden. Die Zuordnung von Sektoren zu Blöcken soll hier nicht interessieren. Der nicht allzu komplizierte Algorithmus ermöglicht es, aufeinander folgende Blöcke ohne größere Zeitverluste zu lesen. Eine einfache Rechnung ergibt, daß eine solche Standard-Diskette für MUTOS eine Kapazität von 494 Blöcken á 512 Byte hat. (Eine Variante mit 8 Sektoren á 512 Byte pro Spur wurde schon für Standard- und Minidisketten erprobt, wird aber noch nicht ausgeliefert.) Für die Formatierung von MUTOS-Disketten muß auf die Formatierungsprogramme eines der Systeme SCP, SIOS oder UDOS zurückgegriffen werden. Das MUTOS-Programm *mkfs* initialisiert auf einer formatierten Diskette ein MUTOS-Filesystem.

Bei Anlauf von MUTOS 8000 muß auf Laufwerk Null eine Diskette mit einem MUTOS-*root file system* eingelegt werden. Auf dieser Diskette sind die Blöcke 250 bis 493 als Swap-Bereich vorgesehen, das heißt als der Speicherbereich, auf den die Prozeß-Umgebung (siehe Punkt 2.1.3.) nicht aktiver Prozesse ausgelagert werden kann.

Damit verbleiben für das eigentliche Wurzel-Filesystem nur 250 Blöcke. Da verschiedene MUTOS-Kommandos Arbeitsdateien unter einem Verzeichnis */tmp* anlegen wollen, müssen ungefähr 100 Blöcke dieser 250 Blöcke als Arbeitsspeicher unbelegt bleiben. In den restlichen Blöcken des Wurzel-Filesystems findet nun nur noch die kleine Zahl von Files Platz, die bei Systemanlauf unbedingt benötigt wird. Neben den schon erwähnten Files /etc/init, /etc/passwd, /etc/ttys, /bin/login und /bin/sh sind das die Kommandos /bin/date (zur Datums- und Uhrzeit- Ein-/Ausgabe), /etc/mount und /etc/umount (zum Montieren und Demontieren weiterer Filesysteme) und ein Kommando zur Überprüfung der Integrität eines File-Systems. Natürlich sind auch die Verzeichnisse /dev (siehe 2.1.2., Spezielle Files) mit ihren speziellen Files, /bin und /etc ebenso wie die leeren Verzeichnisse /mnt, /tmp und /usr auf dem Wurzel-Filesystem zu finden.

Das typische Problem von MUTOS 8000 ist, die Teilmengen des sehr großen Kommando- (Programm-) Satzes so auf einer Diskette unterzubringen, daß ein Nutzer seine Files auf einer dritten Diskette halten und mit ihnen arbeiten kann, ohne daß er ständig die Kommando-Diskette wechseln muß. Denn das unter Punkt 2.1.2. (Montierbare Filesysteme) eingeführte Konzept des hierarchischen Filesystems, das sich über mehrere Datenträger erstreckt, und die ebenfalls in Punkt 2 erwähnte Strategie, Teile von Files und Filesystemdateien in Puffern des Betriebssystem-Kerns zu halten, bedingen, daß vor der Arbeit mit einer Diskette ein *mount*-Kommando gegeben werden muß und nach der Arbeit mit einer Diskette ein *umount* erforderlich ist, soll nicht die Konsistenz des Filesystems auf der Diskette gefährdet werden. Zur Illustration des Problems sei hier erwähnt, daß in voll ausgebauten MUTOS-Systemen ungefähr 5 MByte auf Plattenspeichern für Kommandos (Programme), Systemdateien, Arbeitsspeicher usw. benötigt werden. Dabei sind Nutzer-Daten und Nutzer-Programme nicht berücksichtigt.

Eine Aufzählung der Namen der unter MUTOS 8000 in der Regel vorhandenen speziellen Files soll diesen Punkt abschließen:

/dev/console bezeichnet Tastatur und Bildschirm des BC A 5120.16. Eventuell vorhandene zusätzliche Terminals tragen die Namen /dev/tty0, /dev/ttyl usw.

/dev/lp0 und /dev/lp sind Namen des Drucker-Files. Ein zweiter Drucker wäre unter /dev/lp1 erreichbar. Die Datensenke heißt /dev/null. Den Disketten-Laufwerken 0,1 und 2 sind die File-Namen /dev/rk0, /dev/rk1 und /dev/rk2 zugeordnet. Bei der ebenfalls möglichen „rohen" (ungepufferten) Ein-/Ausgabe auf Disketten sind die entsprechenden Namen /dev/rrk0, dev/rrk1 und /dev/rrk2. Unter den Namen /dev/kmem bzw. /dev/mem sind für privilegierte Nutzer (Programme) die 64 KByte des Betriebssystem-Kerns bzw. der gesamte 256-KByte-RAM erreichbar.

2.4.
Arbeitsspeicheraufteilung

Die 64 K Byte des U 880 sind Arbeitsspeicher des 8-Bit-E/A-Prozessors von MUTOS 8000. Sie sind weder für den MUTOS-8000-Kern, noch für MUTOS-Programme erreichbar. Dieser Speicher wird voll vom E/A-Subsystem des MUTOS 8000 genutzt.
Von den 256 K Byte RAM des U 8001 sind die unteren 64 K Byte für den MUTOS-8000-Kern reserviert. In diesem Bereich befinden sich von Adresse 0 an aufwärts:

– der Programm-Status-Bereich des Prozessors, über den die Unterbrechungsbehandlungen abgewickelt werden
– die in Assembler geschriebenen Teile des Kerns, die vornehmlich zu Beginn und am Ende von Unterbrechungsbehandlungen durchlaufen werden und die zum Beispiel die wenigen Code-Sequenzen enthalten, in denen der Prozessor im segmentierten Modus läuft
– der in C geschriebene Teil des Kerns, Geräte-Treiber eingeschlossen
– globale Systemdaten und Systemdatenbereiche, wie die 512 Byte großen Puffer für die E/A-Arbeit;
– vier Rettebereiche zu je 1 K Byte für auslagerbare Systemdaten von im Hauptspeicher residenten Prozessen (ab Adresse 0EC00H);
– die auslagerbaren Systemdaten des gerade aktiven Prozesses (ab 0FC00H bis 0FFFFH). Teil dieser Daten ist der System-Stackbereich (von 0FFFFH abwärts). Der Prozessor hat zwei Stackregister, eines für die Arbeit im Systemmodus und eines für den Nutzermodus. Bei Prozeß-Wechsel werden auch die Systemstackbereiche der Prozesse ausgetauscht.

Die drei übrigen Segmente zu je 64 K Byte können je ein Programm mit seinen Daten- und Stackbereichen aufnehmen. Für ein Nutzer-Programm, das ja im nichtsegmentierten Modus läuft und demzufolge nur den Adreßraum 0H bis 0FFFFH kennt, obwohl ihm zum Beispiel der physische Adreßraum 20000H bis 2FFFFH zur Verfügung steht, ist der Speicher wie folgt aufgeteilt:

– Ab Adresse 00000H beginnen die Befehle des Programms. Nach einem execute-Systemruf geht die Steuerung an den Befehl mit der Adresse Null.
– An die Befehle schließen sich die nach dem Programm-Verbinderlauf initialisierten Daten des Programms an.
– Auf die initialisierten Daten folgen die nichtinitialisierten Datenbereiche, die jedoch vom Betriebssystem beim Laden des Programms auf binär Null gelöscht werden.
– Ab Adresse 0FFFFH abwärts erstreckt sich der Stackbereich des Programms. Die bei einem execute-Ruf an das Programm übermittelten Argumente werden beim Laden im Stack abgelegt. Das Stack-Register zeigt beim Start des Programms auf das letzte (unterste) Wort des belegten Stackbereiches, das die Argument-Zahl enthält. An dieses Wort schließen sich aufsteigend Zeiger auf die Argument-Zeichenketten an.

Beim Auslagern von Prozessen wird der Bereich zwischen dem Ende der nichtinitialisierten Daten und dem Wort, auf welches das Stackregister zum Zeitpunkt des Auslagerns zeigt, nicht in den Swap-Bereich übernommen. Ein Programm kann nur dann gefahrlos dynamisch mehr Speicher, als ihm beim Laden zugewiesen wurde, benutzen, wenn es entsprechende Wünsche mit einem Systemruf an das Betriebssystem übermittelt. Eine dynamische Ausweitung des Stackbereiches ist hingegen unproblematisch.

2.5.
Kommandosystem

Unter diesem Punkt sollen sowohl Shell, der Kommando-Interpreter von MUTOS, als auch ein Basissatz von Kommandos eingeführt werden.

2.5.1.
Shell – der Kommandointerpreter

Der Nutzer verkehrt mit dem System über einen Kommandointerpreter. Wenn es auch unter MUTOS für jeden Nutzer einen eigenen Kommandointerpreter geben kann, ist es doch wichtig, den Standard-Interpreter Shell zu kennen.
Für Shell besteht die einfache Form einer Kommando-Zeile aus einem Kommando-Namen, dem durch Leerzeichen getrennte Argumente folgen können:
command arg1 arg2 ... argn.
Shell erzeugt aus einer solchen Eingabezeile Zeichenketten. Dann wird ein File mit dem Namen **command** gesucht. Der Kommando-Name ist ein üblicher Pfad-Name. Beginnt er mit „/", so ist der Suchweg klar. Bei unvollständigen Namen beginnt die Suche im laufenden Verzeichnis. Führt sie zu keinem Erfolg, wird vor den Kommando-Namen ein Präfix wie /bin oder /usr/bin gestellt. (Welche Präfixe in welcher Reihenfolge benutzt werden, wird durch die vom Kommandointerpreter geführte PATH-Variable festgelegt. Sie kann, wie auch andere Variable, vom Nutzer modifiziert werden). Wird das File gefunden, erzeugt Shell mit einem fork-Ruf eine Kopie von sich selbst. Im Eltern-Prozeß wartet Shell auf die Beendigung des Kind-Prozesses. Ist sie erfolgt, zeigt Shell mit einem Prompt-Zeichen die Bereitschaft an, eine neue Kommando-Zeile entgegenzunehmen.
Im Kind-Prozeß sammelt Shell die Kommando-Argumente ein und ruft mit
execute (command, arg1, arg2, ..., argn)
das gefundene File zur Ausführung auf.
Daraus folgt, daß unter MUTOS alle ausführbaren Programme Kommandos sind. Shell selbst ist ebenfalls ein ausführbares Programm. Nur eine kleine Zahl von Kommandos wird von Shell ausgeführt, ohne einen Kind-Prozeß zu erzeugen. Das schon vorgestellte **cd**-Kommando ist eines von ihnen. Die von Shell interpretierte Kommandosprache ist durchaus mit höheren Programmiersprachen vergleichbar. (Sie kennt z. B. bedingte Anweisungen, while- und for-Schleifen, case-Anweisungen usw.). Hier sollen jedoch nur einige Basiskomponenten dieser Sprache eingeführt werden.

Standard-Ein/Ausgabe

Jedes Programm, das durch Shell zur Ausführung gebracht wird, findet, wenn es die Steuerung bekommt, schon drei eröffnete Files mit den Deskriptoren 0, 1 und 2 vor. Das mit Deskriptor 1 ansprechbare File ist für Ausgabe eröffnet. Es ist am besten als standard-output zu deuten. Gewöhnlich ist es das Nutzerterminal.
Deskriptor 0 betrifft ein File, das für Lesen eröffnet ist. Es ist als standard-input zu deuten und weist meist ebenfalls auf das Nutzerterminal. So kann ein Pro-

gramm ohne open-Rufe Nachrichten über File-Deskriptor 1 ausgeben und über File-Deskriptor 0 Zeichen einlesen, die vom Nutzer eingegeben werden.
Shell kann diese Standardzuordnung zum Nutzerterminal ändern. Ist ein Element einer Kommando-Zeile von der Form >**argumentname**, so weist für die Dauer der Kommandoabarbeitung Deskriptor 1 nicht auf das Terminal, sondern auf ein File mit dem Namen **argumentname**. Zum Beispiel werden mit **ls** >**Verzeichnis** die Namen der im laufenden Verzeichnis eingetragenen Files in das File **Verzeichnis** gebracht. Das File **Verzeichnis** wird von Shell im Kind-Prozeß mit einem create-Ruf erzeugt, bevor das Kommando **ls** ausgeführt wird.
Analog dazu wird mit einem Argument <**filename** bewirkt, daß das aufgerufene Kommando nicht das Nutzer-Terminal, sondern ein File **filename** anspricht, wenn es über File-Deskriptor 0 Zeichen eingibt. Es sei noch einmal betont, daß Argumente der Formen <**namex** oder >**namey** von Shell interpretiert werden. Das gerufene Programm bekommt sie nicht zu sehen.
File-Deskriptor 2 weist ebenfalls wie File-Deskriptor 1 auf die Terminal-Ausgabeeinrichtung. Das Umlegen des standard-output wirkt sich jedoch nicht auf diesen Deskriptor aus, so daß Fehlermeldungen auf das mit Deskriptor 2 verbundene File an das Terminal gehen. Natürlich kann jedes Programm die zu den Deskriptoren 0, 1 oder 2 gehörenden Files schließen und eventuell an ihrer Stelle andere Files eröffnen. Programme, die nicht von Shell aktiviert werden, können also nicht unbedingt mit der Standard-Ein-/Ausgabe rechnen.

Filter

Als Filter werden Programme (Kommandos) bezeichnet, die nur ihren standard-input lesen, transformieren und wieder auf den standard-output ausgeben. Shell bietet nun eine Notation, mit deren Hilfe der standard-output eines Programms zum standard-input eines anderen Programms gemacht werden kann: Eine Zeile, bestehend aus einer Folge von Kommandos (mit ihren Argumenten), die durch „|" getrennt sind, veranlaßt Shell, die Kommandos parallel auszuführen und mit Hilfe von pipes den standard-output eines Kommandos zum standard-input des nächsten Kommandos zu machen. Als Beispiel sei folgende Sequenz betrachtet:
ls|pr-2|opr
ls listet den Inhalt des laufenden Verzeichnisses auf. **pr** ordnet seinen standard-input seitenweise und versieht ihn mit einer Überschrift, die das Datum enthält. (Das Argument -2 veranlaßt **pr** zur zweispaltigen Ausgabe). **opr** spoolt seinen standard-input zu einem offline-Drucker.
Zweifelsohne ist das eine elegantere Methode, als alle diese Funktionen in einem Programm zu vereinen oder die folgenden vier Kommandos einzugeben, die Shell auch verarbeiten könnte:

```
ls >temp1
pr -2 <temp1 >temp2
opr <temp2
rm temp1 temp2
(Mit dem letzten Kommando werden die Zwischendateien gelöscht!)
```

Kommando-Seperatoren, Hintergrundarbeit und Kommandofiles

Kommandos müssen nicht auf getrennten Zeilen stehen. Zwei durch Semikolon getrennte Kommandos (mit ihren Argumenten) werden von Shell nacheinander abgearbeitet.
Wird ein Kommando von einem **„&"** gefolgt, so wartet Shell im Eltern-Prozeß nicht auf die Beendigung des im Kind-Prozeß angestoßenen Kommandos, sondern teilt dem Terminal-Nutzer nur den Prozeß-Identifikator des Kind-Prozesses mit, bevor erneut Eingabebereitschaft angezeigt wird.
Zum Beispiel wird mit
as source& ls files&
sowohl ein Assembler im Hintergrund angestoßen als auch das Inhaltsverzeichnis parallel zur laufenden Terminalarbeit des Nutzers erzeugt.
Shell ist selbst ein Kommando und kann rekursiv gerufen werden. Eine Kommando-Zeile der Form
sh <command-file
bewirkt, daß im Kind-Prozeß das Programm Shell abgearbeitet wird, dessen standard-input kein Terminal, sondern eine File ist, das Kommando-Zeilen enthält.
Kommandofiles müssen jedoch nicht auf diese umständliche Weise ausgeführt werden. Meldet nämlich das System im Shell-Kind-Prozeß, daß das mit **execute** angestoßene File zwar kein ausführbares Programm, wohl aber ein File mit Ausführberechtigung ist, so vermutet Shell, daß es sich um ein Kommandofile handelt, legt den eigenen standard-input auf dieses File und beginnt normale Shell-Kommando-Interpretationen. Das heißt, gewöhnliche Files, die Kommando-Zeilen enthalten und bei denen ein passendes Ausführ-Berechtigungsbit eingeschaltet ist, können vom Terminal wie gewöhnliche Programme aufgerufen werden. Weitere Shell-Leistungen wie Parameter-Substitution, Erzeugung von Filenamen-Argumenten mittels eines ausgefeilten pattern matching in Verzeichnissen usw. sollen hier nur erwähnt werden.

2.5.2.
Kommandos von MUTOS 8000

Die im folgenden vorgestellten 46 Kommandos bilden den Basissatz an MUTOS-8000-Kommandos. Sie befinden sich entweder auf dem Wurzel-Filesystem oder auf einer von zwei Kommandodisketten. Von den beiden Kommandodisketten ist eine ausschließlich dem Problemkreis Editieren, Kompilieren, Assemblieren und Verbinden gewidmet. Diese Diskette enthält auch die Standard-Bibliothek mit ihren 136 Objektmoduln, die der Programmierer bei Bedarf in seine Programme einbeziehen kann. Bei den Erstanwendern von MUTOS 8000 liegen weitere MUTOS-Programme (Kommandos) und eventuell zugehörige Bibliotheken vor, auf sie soll hier nur verwiesen werden.
Wie auch in anderen Betriebssystemen üblich, kann die Ausführung von MUTOS-Kommandos durch eine Vielzahl von Flags beeinflußt werden. Sie zu erläutern bleibt dem Systemhandbuch vorbehalten. Auch auf die vielfältigen Möglichkeiten, die sich durch Shell-Leistungen, wie Umlegen der Standard-Ein-/Ausgabe und Pipelines, ergeben, kann hier nicht eingegangen werden.

adb Ein interaktiver Debugger, der den Online-Test beliebiger Programme, die Analyse von Speicherabzügen abgestürzter Programme und die Modifikation beliebiger Files gestattet. Er enthält einen Reassembler.

ar Der Archivar. Er dient dem Anlegen und der Pflege von Bibliotheken. Eine Bibliothek gestattet das speichereffiziente Zusammenfassen von vielen Files in einem File.

as Der Assembler. Die vom Assembler akzeptierte Sprache ist eine Teilmenge von PLZ/ASM. Der Assembler erzeugt Objektmoduln.

cat Concatenate. Das Kommando erzeugt aus einem oder mehreren Argument-Files ein Ausgabefile, das auf den Standard-Output geschickt wird. Es wird auch zum Betrachten von Files benutzt.

chmod Dient dem Ändern der Schutzbits eines oder mehrerer Files (siehe 2.1.2., Das Schutzsystem)

chown Die Nutzer- und Gruppenidentifikation der als Argumente angegebenen Files kann mit diesem Kommando geändert werden.

clri Ein Kommando für den System-Manager. Es löscht inodes (siehe 2.1.2., Implementierung)

cmp Compare. Vergleicht zwei Files und listet die erste oder alle Differenzen aus.

convert Konvertiert ein MUTOS-8000-Filesystem auf einer Diskette in ein MUTOS-1630-Filesystem (und umgekehrt). Die beiden MUTOS-Filesysteme unterscheiden sich, weil die Byte-Reihenfolge in einem 16-Bit-Wort auf beiden Rechnern unterschiedlich ist.

cp Kopiert Files (stellt Duplikate her).

cc C-Compiler-Kommando. Unter seiner Steuerung werden automatisch C-Quellprogramme compiliert, assembliert und unter Heranziehung der Standard-Bibliothek zu einem ausführbaren Programm verbunden. Eine Vielzahl von Flags gestattet, diesen Vorgang zu beeinflussen. Das Kommando akzeptiert neben C-Programmen auch Assemblerquellen und Objektmoduln als Argumente. Es ruft in Kindprozessen auch das as- und ld-Kommando auf.

cd Change current directory. Wechselt das aktuelle Verzeichnis.

date Setzen und Anzeige der Systemzeit.

dcheck Ein Kommando zum Überprüfen der Konsistenz der Verzeichnisse von Filesystemen.

df Zeigt die Zahl freier Blöcke in Filesystemen an.

dd Konvertieren und (evtl. partielles) Kopieren von Files.

echo Gibt seine Argumente auf den Standard-Output aus.

ed Ein außerordentlich leistungsfähiger, zeilenorientierter Editor.

icheck Prüft die Konsistenz eines Filesystems und zeigt wichtige Kenngrößen an. Kann zum Aufbau einer neuen Freispeicherliste genutzt werden.

kill Beenden eines Prozesses.

ld Der Programmverbinder. Wandelt Objektmoduln in ausführbare Programme um. Zur Auflösung von Referenzen durchsucht er als zusätzliche Argumente angegebene Objektmodul-Bibliotheken.

ln link. Erzeugt links, das heißt, ordnet existierenden Files weitere Namen zu.

login Anmeldung beim System. Durch Eingabe eines login-Kommandos wird auch eine vorangegangene Sitzung abgeschlossen.

ls Listet mehr oder weniger ausführlich den Inhalt von Verzeichnissen auf dem Standard-Output aus.

mkdir Make directory. Dient der Einrichtung von Verzeichnissen.

mkfs Make file system. Richtet auf einer formatierten Diskette ein Filesystem ein.

mknod Legt ein spezielles File (special file) in einem Filesystem an. Das mit einem speziellen File zu erreichende Gerät muß allerdings zuvor im Betriebssystem-Kern verankert sein; die entsprechenden Tabellen-Eintragungen und Treiber müssen dort existieren.

mount Gliedert ein auf einer Diskette existierendes Filesystem in die zugreifbare Filehierarchie ein.

mv Move. Dient der Umbenennung von Files. Es transportiert zum Beispiel einen Filenamen aus einem Verzeichnis in ein anderes.

ncheck Ermittelt die zu inode-Nummern gehörenden Filenamen bzw. listet alle Filenamen eines Filesystems auf dem Standard-Output aus.

nm Gibt nach verschiedenen Kriterien geordnet die Symboltabelle von Objekt- oder Lademoduln aus.

od Octal dump. Gibt ein File oder einen Teil eines Files oktal, hexadezimal oder in einem anderen Format aus.

passwd Ändert das Passwort eines Nutzers bzw. gestattet einem Nutzer, seinen login-Namen erstmalig durch ein Passwort zu schützen.

pr Dient der formatierten Ausgabe eines oder mehrerer Files auf dem Standard-Output. Mit Hilfe von Flags können zum Beispiel mehrspaltige Ausgabe, die Ausgabe von Überschriften, die Ausgabebreite usw. angefordert werden.

ps Informiert über den Status aller im System existierenden Prozesse.

pstat Mehr oder weniger detaillierte Ausgabe von internen System-

informationen (nur für Systemprogrammierer von Interesse).

pwd Print working directory. Zeigt das aktuelle Verzeichnis an.

rm Löscht Filenamen aus Verzeichnissen und bewirkt damit eventuell das Löschen von Files (siehe 2.1.2., Verzeichnisse).

rmdir Löscht die Namen von Verzeichnissen und damit eventuell auch Verzeichnisse. Das Löschen wird nur durchgeführt, wenn ein Verzeichnis nur noch die beiden Namen „." und „.." enthält.

sh Der Kommando-Interpreter. Als Kommando wird Shell gewöhnlich mit gewissen Flags aufgerufen, die es gestatten, komplizierte Kommando-Prozeduren zu testen.

size Gibt für Objekt- und Lademoduln die Anzahl der Bytes an, die vom Befehlsteil, vom Teil mit initialisierten Daten und vom Teil mit nichtinitialisierten Daten benötigt werden.

strip Entfernt aus Objektmoduln Verschieblichkeitsinformationen und die Symboltabelle, aus Lademoduln die Symboltabelle.

stty Setzt Terminal-Optionen. So kann zum Beispiel mit Hilfe von stty der Terminal-Treiber bewogen werden, bei der Ausgabe von Files nach jeweils 24 Zeilen die Ausgabe zu unterbrechen und erst nach Eingabe eines beliebigen Zeichens fortzusetzen.

sync Aktualisiert die Superblöcke der Filesysteme und bewirkt die Ausgabe sämtlicher im Systemkern gepufferter Datenblöcke. Das Kommando muß vor Abschalten des Rechners gegeben werden.

wc Word count: Zählt Zeilen, Wörter und Bytes in Files.

who Listet im Mehrnutzer-Betrieb die login-Namen der normalen aktiven Nutzer auf.

2.6. Logisches und physisches E/A-System

Ein Teil der System-Rufe, die im Nutzerprogramm zur Arbeit mit Files benutzt werden können, wurde schon unter Punkt 2.1.2. (Ein-/Ausgabe-Systemrufe) eingeführt. An dieser Stelle sollen nun nicht die restlichen Rufe nachgereicht werden, sie sind bei Bedarf dem Systemhandbuch zu entnehmen.
Für Nutzerprogramme sind die Unterschiede zwischen der Arbeit mit gewöhnlichen Files und der Arbeit mit speziellen Files unwesentlich. (Dies sind die MUTOS-Entsprechungen für logische und physische E/A). Für beides werden die gleichen Systemrufe benutzt. Erwähnenswert ist jedoch, daß zum Beispiel das spezielle File /dev/rk0, also die Diskette auf Laufwerk Null, 494×512 Bytes enthält (nämlich die 512-Byte-Blöcke 0 bis 493), die durch entsprechende read-Rufe nach einem vorausgegangenen open-Ruf nacheinander byteweise eingelesen werden können. Dies ist möglich, weil auch in diesem Fall die E/A-Arbeit des Systems gepuffert erfolgt. Wird jedoch zum Beispiel auf Laufwerk Null mit Hilfe des speziellen Files /dev/rrk0 zugegriffen, so erfolgt die E/A-Arbeit ungepuffert, und das System akzeptiert nur read-Rufe, die Vielfache von 512 Bytes anfordern (maximal 4 K Byte). Mit diesem Beispiel soll auf eine weitere Klassifizierung spezieller Files hingewiesen werden, die unter MUTOS üblich ist. Da aber bei MUTOS 8000 die Diskette der einzige Gerätetyp ist, bei dem sowohl die betriebssysteminterne Pufferung von 512-Byte-Blöcken als auch die ungepufferte Arbeit unterstützt werden, soll die Unterscheidung in sogenannte *block special files* und *character special files* hier nur erwähnt werden, zumal die von UNIX stammende Namensgebung ziemlich unglücklich ist.
Die Abbildung logischer Ein-/Ausgaben, das heißt von Ein-/Ausgaben für gewöhnliche Files, auf physische Ein-/Ausgaben, die direkten Gerätezugriffe, erfolgt bei MUTOS im Betriebssystemkern. Ein-/Ausgaben auf spezielle Files führen bei MUTOS in der Regel ebenfalls erst auf zwischengeschaltete Programme, ehe die eigentlichen Gerätetreiber aktiviert werden. Die Treiber im MUTOS-8000-Kern bereiten die physischen Ein-/Ausgaben etwa in dem Umfang vor wie die MUTOS-1630-Treiber. An die Stelle der Modifikation von Gerätesteuereinheiten-Registern beim MUTOS 1630 tritt bei MUTOS 8000 die Kommunikation mit dem E/A-Subsystem, das in den 64 K Byte des U-880-Systems residiert. Die Gerätetreiber, die im E/A-Prozessor die physischen Ein-/Ausgaben realisieren, sind zur Zeit UDOS-Treiber oder aus UDOS-Treibern abgeleitete Routinen. Die Hinzunahme neuer Gerätetypen macht bei MUTOS 8000 Anpassungsarbeiten im E/A-Subsystem und im MUTOS-Kern erforderlich. Die nötigen Vorbereitungen dafür sind getroffen worden.

edv aspekte

lieferbar

4/1985

PASCAL-Anwendungen
für Wirtschaft/Wissenschaft/Technik

Geboten wird eine Auswahl von Aufgaben, Programmen und Resultaten als breites Spektrum von Möglichkeiten für Vergleiche der Leistungsfähigkeit von PASCAL-Implementationen auf Rechnern unterschiedlichster Leistungsfähigkeit. Ein weiteres Anliegen der Sammlung besteht darin, eine Unterstützung für die Einarbeitung in die Programmierung mit PASCAL zu liefern. Deshalb wurden nicht nur für alle Programme die zugehörigen Resultate angegeben, sondern die Quelltexte mit ausführlichen Kommentaren versehen.

Bestellungen richten Sie bitte an den Verlag Die Wirtschaft, Abt. Vertrieb, 1055 Berlin, Am Friedrichshain 22.

Das Betriebssystem SIOS

Dr. Volkmar Köhler, Hans-Gerd Unterschütz
VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt

3. SIOS

3.1. Systemübersicht

Das Betriebssystem SIOS 1526 wurde als erstes Betriebssystem für die Bürocomputer A 5120 und A 5130 sowie die Terminals K 8924 und K 8927 geschaffen. Aus diesem Grunde besitzt es eine weite Verbreitung, und die unter SIOS abarbeitbare Software hat einen beachtlichen Umfang angenommen.
SIOS 1526 ist für folgende *Haupteinsatzgebiete* prädestiniert:
- Massendatenerfassung und -vorverarbeitung
- Datenübertragung und Terminaleinsatzfälle
- Buchung, Fakturierung und Abrechnung.

Darauf zugeschnitten sind die *Mittel zur Entwicklung* von Programmen in folgenden Programmiersprachen:
– CPU-Assembler für U 880 → CABS
– Makroassembler MABS und Systemassembler SABS
– BASIC
– kommerzielles PASCAL
– COBOL.

SIOS bietet von allen Bürocomputer-Betriebssystemen die umfangreichste Hardware-Unterstützung. Es ist für die gesamte zum Bürocomputer-Bestand gehörende *Hardware* verfügbar. Dazu gehört:
– ZRE K 2526 mit 64 KByte Arbeitsspeicher
– Speichererweiterung auf 112 KByte
– 1 ... 4 Laufwerke Floppy-Disk (MF 3200, MF 6400, K 5602.10, K 5602.20, K 5600.10, K 5600.20)
– Tastatur K 7634, K 7636, K 7637 (auch alte Versionen K 7604, K 7606)
– Monitor K 7221 und K 7222
– Drucker SD 1152, 1157, K 6311 sowohl mit PIO- als auch mit IFSS-Interface (auch SD 1156, 1154 möglich), Betrieb mit Haupt- und Zusatzdrucker möglich
– Datenfernübertragung über V.24 oder IFSS
- asynchrone Übertragung analog AP 62/64
- synchrone Übertragung analog BSC I und III
- prozedurfreie Übertragung

– 1 ... 2 Laufwerke Kassettenmagnetbandgerät K 5200
– ½"-Magnetband CM 5300
– Schreib-Lese-Einheit (SLE) K 6501, automatische Lese-Einheit (ALE) K 6502, Hand-Lese-Einheit (HLE)K 6503 für Plastkarte mit Magnetstreifen
– Lochbandgerät K 6200.

In praktisch relevanten Fällen wird jedoch immer nur eine Untermenge von diesem Hardwareangebot an einer konkreten Maschine bedient werden müssen, so daß im Betriebssystem nur die entsprechend der Hardware erforderlichen Module bereitgestellt werden müssen. Zu diesem Zweck muß das Betriebssystem für jede konkrete Hardware generiert werden. Dafür steht das Dienstprogramm SGEN mit der Moduldatei SYSORGxx zur Verfügung (xx = Versionsnummer). Der residente Teil von SIOS ist entsprechend Abb. 1 strukturiert. Er besteht aus den beiden Komponenten
- SIEX

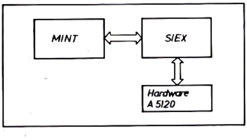
das eigentliche Steuerprogramm für die gerätenahen Funktionen (sogenannte physische Module) und Betriebssystemsteuerroutinen
- MINT

der Makrobefehlsinterpreter für die Makroassemblersprache MABS; er realisiert neben der Makrobefehlsausführung die gesamte Dateiorganisation und logische E/A-Arbeit.
Für jedes E/A-Gerät gibt es in SIEX einen physischen und in MINT einen logischen Modul. Diese beiden Teile bedingen einander und können nicht getrennt werden. Modularität im Sinne von SIOS bedeutet deshalb, daß beim Systemgenerieren nur jene Module aus SIEX und MINT in das Betriebssystem einbezogen werden, für die auch die entsprechenden Hardwarekomponenten körperlich vorhanden sind. Es ist aber nicht möglich, die Module von SIEX zu benutzen und die entsprechenden von MINT nicht zu generieren, selbst wenn vorausgesetzt werden kann, daß die Makrobefehlssprache MABS bei dem konkreten Anwendungsfall nicht benutzt wird.

3.2. Systeminitialisierung

Zunächst muß bemerkt werden, daß SIOS ein sehr variables Initialisierungskonzept besitzt. Es werden drei prinzipielle Varianten unterschieden:
① *ROM-Variante*
SIEX und MINT sind speicherresident in ROM/PROM-Bausteinen untergebracht und beim Einschalten der Anlage sofort verfügbar. Mit dem durch das Netzteil beim Einschalten erzeugten RESET-Signal für den Rechner werden sofort die sogenannten Betriebsbeginnroutinen im Grundmodul und in den E/A-Modulen des Betriebssystems abgearbeitet. Dabei wird im Betriebssystem eine definierte Ausgangsstellung erzeugt, und alle E/A-Geräte werden auf ihren spezifischen Anfangswert synchronisiert. Anschließend erscheint auf dem Dialoggerät (in der Regel der Bildschirm) die Meldung:
MONITOR – SIOS 1526/0.8 (M)_
Damit wird Eingabebereitschaft für Betriebssystemkommandos signalisiert und die Systeminitialisierung ist abgeschlossen.
② *RAM-Variante*
Im Arbeitsspeicherbereich des Bürocomputers existiert kein Festwertspeicher für die Aufnahme des Betriebssystems. Da die eingesetzten RAM-Bausteine beim Abschalten der Spannung ihre Informationen verlieren, muß das Betriebssystem (genauer: der residente Teil davon) nach jedem Einschalten neu in den Arbeitsspeicher des Rechners geladen werden. In diesem Falle befindet sich das System immer auf einem Datenträger (beim Bürocomputer sind dafür Diskette bzw. Kassette geeignet).

Abb. 1 Systemkomponenten des residenten Teils von SIOS



Dieser Vorgang ist automatisiert worden und läuft folgendermaßen ab:
Mit dem RESET-Signal des Netzteiles wird der Anfangslader (siehe 1.4.1.) aktiviert. Er lädt seinen Speicherinhalt in den RAM und schaltet danach den PROM ab, damit der volle Adreßraum von 64 KByte für RAM zur Verfügung steht. Dieses Programm im RAM sucht nun auf den angeschlossenen Datenträgern nach einem Systemlader (siehe 1.4.2.), der sich immer auf definierten Stellen auf der Systemdiskette bzw. Systemkassette befinden muß:
– Diskette: Spur 0, Sektoren 1 ... 4
– Kassette: Blöcke 0 ... 3.
Der Systemlader wird nun seinerseits in den RAM geladen und angesprungen. Er lädt sodann die Betriebssystemdatei in den RAM und springt den Grundmodul an. Von jetzt an wird der Ablauf mit dem unter ① skizzierten identisch.
Wichtig ist nun noch zu bemerken, daß die Systemdatei auf der Diskette immer mit der Spur 1 beginnt. Sie muß den Namen SYSARExx (xx = Versionsnummer) besitzen. Ihre Länge (Anzahl der Sektoren) ist im Systemlader als Konstante enthalten. Auf der Systemdiskette können dann entsprechend der noch freien Diskettenkapazität weitere Daten abgespeichert werden. Im Unterschied dazu darf eine Systemkassette keine weiteren Kassettendateien enthalten, da das Betriebssystem hier in Form einer Datei im sogenannten SIMPLE-Format aufgezeichnet ist (siehe 3.3.3.).

③ *Nachladevariante*
Der Arbeitsspeicher des Bürocomputers ist modular konfigurierbar, das heißt, er kann in festen Schrittweiten auf- oder abgerüstet werden. Bei RAM-Steckeinheiten sind 4 KByte bzw. 16 KByte typische Werte, bei ROM 16 KByte. Wenn nun z. B. bei einer ROM-Maschine (Variante ①) das Betriebssystem eine Größe erreicht, die knapp über den Modulgrenzen liegt (beispielsweise 18 KByte), dann wird die verfügbare ROM-Kapazität bzw. die Adreßkapazität überhaupt ausgesprochen ungünstig ausgenutzt. In einem solchen Falle ist eine Nachladevariante angezeigt. Hier werden nur so viel ROM-Module eingesetzt, wie tatsächlich vollständig genutzt werden. Der darüber hinausgehende Teil des Betriebssystems befindet sich als Nachladesystem auf einer Nachladediskette bzw. Nachladekassette, die außerdem einen sogenannten Nachlader (SYC) enthalten muß.
Im ROM-Teil des Betriebssystems müssen sich neben dem Grundmodul mindestens noch die Module für Tastatur, Bildschirm und der physische Modul von Floppy-Disk bzw. Kassette befinden. Die Inbetriebnahme erfolgt nun analog der unter ① genannten Variante. Nachdem sich das System mit der Monitorzeile gemeldet hat, wird mit dem Monitorkommando SYC ein geschlossenes Nachladen aller auf dem Nachladedatenträger befindlichen Systemmoduln veranlaßt. Damit ist jetzt das gesamte Betriebssystem im Arbeitsspeicher, wovon aber ein Teil im ROM und ein weiterer im RAM steht. Nunmehr erst ist die Bedienung aller zur Hardware gehörenden E/A-Geräte gewährleistet. Das Betriebssystem meldet sich wieder mit der Monitorzeile.

3.3. Dateiarbeit mit Disketten und Kassetten

Obwohl international bei den magnetischen Datenträgern der Trend eindeutig zu Disketten und Festplatten in Winchester-Technologie hinläuft, besitzen für bestimmte Anwendungsfälle Magnetbandkassetten noch eine beachtliche Bedeutung. Aus diesem Grunde wurden sie für die Bürocomputer optional bereitgestellt. Über das Betriebssystem SIOS wurde auch hier eine Dateiverwaltung realisiert. Sowohl bei der Diskette als auch bei der Kassette wurden Aufzeichnungsverfahren und Dateiorganisation streng an internationale Normen angelehnt. Es sind dies bei:
– Diskette:
ISO TC 97 Sc 11 Nr. 209 (KROS 5108/01 bzw. KROS 5110/01)
– Kassette:
ISO TC 97 Sc 11 Nr. 35 (KROS 5109).
Diese Normen enthalten neben einer technisch-physikalischen Beschreibung des Datenträgers die Vorschriften für die physische und logische Einteilung des Speichermediums.

3.3.1. Diskettenaufbau

Zunächst soll kurz auf den physischen Aufbau eingegangen werden. Die Diskette ist in einzelne, konzentrische Spuren eingeteilt. Die Anzahl der Spuren kann Tab. 1 entnommen werden. Die Spur 0 wird als Indexspur bezeichnet. Sie verfügt unabhängig von der sonstigen Aufteilung der Diskette immer über 26 Sektoren zu je 128 Byte. Damit kann die Indexspur von jedem Floppy-Disk-Laufwerk des Bürocomputer-Bestandes gelesen werden. Aus den Informationen der Indexspur läßt sich dann feststellen, auf welchem Laufwerkstyp die Diskette initialisiert wurde.
Von der Spur 1 an beginnt der Bereich für die Datenaufzeichnung. Hier sind je nach Initialisierung mehrere Sektorformate möglich. Sie sind ebenfalls in Tab. 1 zusammengestellt. Die Länge aller physischen Sektoren einer Diskette (außer Spur 0) ist immer gleich.
Die jeweils letzten beiden Spuren einer jeden Diskette sind die sogenannten Ersatz- oder Austauschspuren. Werden nämlich bei der Initialisierung der Diskette fehlerhafte Spuren bemerkt, so werden diese für eine Datenaufzeichnung gesperrt und dafür die Ersatzspuren zugewiesen. Allerdings dürfen maximal zwei Spuren fehlerhaft sein, da nur zwei Ersatzspuren zur Verfügung stehen. Bei mehr als zwei fehlerhaften Spuren ist die Diskette für SIOS unbrauchbar. Das gleiche gilt, wenn die Indexspur fehlerhaft ist. Zur Initialisierung einer Diskette wird das Dienstprogramm INIT verwendet.

3.3.2. Diskettendateien

Die Indexspur ist die Beschreibungsspur für Diskette und Dateien. Auf ihr befinden sich bei SIOS alle Kennsätze. Tab. 2 zeigt den prinzipiellen Aufbau der Indexspur. Dateien selbst können sich dann auf der Diskette erst ab Spur 1 befinden. Aus der maximal möglichen Anzahl von Dateikennsätzen (19) ergibt sich, daß je Diskette maximal 19 Dateien untergebracht werden können. Alle Kennsätze sind auf Grund des

physischen Aufbaus der Indexspur immer 128 Byte lang. Logisch werden bei ihnen jedoch immer nur 80 Byte benutzt, die Positionen 81 ... 128 sind mit NUL (0H) aufgefüllt. Im folgenden sollen die Kennsätze kurz beschrieben werden. Für Detailinformationen ist die KROS 5108 bzw. 5110 heranzuziehen.

Fehlerkennsatz (ERMAP)

Der Fehlerkennsatz dient der Identifikation fehlerhafter Spuren auf der Diskette, die bei der Initialisierung erkannt wurden.

Datenträgerkennsatz (VOL1)

In diesem Kennsatz sind Informationen über den physischen Diskettenaufbau und den Eigentümer der Diskette enthalten. Es sind u. a. folgende Informationen aufgezeichnet:

- Datenträgername (6 Zeichen)
- Eigentümername (14 Zeichen)
- Datenträgerart (Standard- oder Minidiskette)
- Bitdichte (einfach oder doppelt)
- physische Sektorlänge (128, 256, 512 oder 1024 Byte)
- physische Satzfolge (Sektornumerierung innerhalb der Spuren).

Fehlerkennsatz und Datenträgerkennsatz werden bereits bei der Initialisierung angelegt und entsprechend den Bedieneranweisungen bzw. mit Standardwerten ausgefüllt. So wird z. B. durch das Dienstprogramm INIT im Datenträgerkennsatz immer der Datenträgername ROBINI (Robotron-Initialisierung) eingetragen.

Dateikennsätze (HDR1)

Je Datei ist ein Dateikennsatz erforderlich. Da damit die Anzahl der Dateien auf einer Diskette stark beschränkt ist, werden Programme in SIOS nicht als Dateien, sondern als Bücher einer Bibliothek aufgezeichnet. Eine Bibliothek besitzt dann Dateistatus. Das ist ein wesentlicher Unterschied zu den beiden anderen 8-Bit-Betriebssystemen UDOS und SCP, wo jedes Programm als Datei behandelt wird.

Ein Dateikennsatz dient der Identifikation der Datei und beschreibt ihre Anordnung und Größe auf dem Datenträger. Der Aufbau eines Dateikennsatzes ist in Tab. 3 dargestellt. Als wichtigste Informationen sind enthalten:

- Dateiname (bei SIOS maximal 8 Zeichen lang)
- Anfang des Dateibereiches (BOE)
- Ende des Dateibereiches (EOE)
- Ende der Daten im Dateibereich (EOD)
- Schreibschutz
- Datenaustauschniveau
- Mehrdatenträgeranzeige und Datenträgerfolgenummer
- logische Satzlänge
- Dateiorganisation (sequentiell, nicht sequentiell).

Nachfolgend soll noch ausführlicher auf die Varianten der Dateiorganisation unter SIOS eingegangen werden, da zwischen der Dateiorganisation und dem konkreten Anwendungsfall oft eine enge Beziehung besteht.

In SIOS muß sich eine Datei physisch als zusammenhängender Bereich auf einer Diskette, im Falle von Multi-Volume-Dateien über mehrere Disketten, erstrecken. Ebenso ist es wichtig, eine Datei mit dem Dienstprogramm FGEN auf Diskette einzurichten, bevor auf sie das erstemal zugegriffen werden kann. Die Datei muß sozusagen außerhalb des Anwenderprogrammes „vordefiniert" werden, wobei ihre Grenzen mittels FGEN bereits festgelegt werden.

Bezüglich der Zugriffsmethoden (Abb. 2) unterstützt SIOS drei Varianten:

Durchmesser	8"	8"	5,25"	5,25"
Aufzeichnungsverfahren	FM	MFM	MFM	MFM
Spuranzahl	77	77	40	80 (77 genutzt)
Spur 0:				
Sektoranzahl	26	26	26	26
Sektorlänge	128	128	128	128
alle anderen Sektoren ab Spur 1: Anzahl/Länge	26/128 15/256 8/512 4/1024	44/128 26/256	26/128 15/256 8/512 4/1024	26/128 15/256 8/512 4/1024

Tab. 1 Wesentliche physische Kennwerte von Disketten

a) Sequentiell organisierte Datei

Die Datensätze (records) stehen lückenlos aufeinanderfolgend ab BOE auf Diskette. Der Zugriff auf die Sätze erfolgt bezüglich der Satznummer nur in aufsteigender oder in absteigender Reihenfolge. Der erste und der letzte Satz sind „direkt" positionierbar. EOD gibt den nächsten freien Satz innerhalb des Dateibereiches (zwischen BOE und EOE) an.

Bei Multi-Volume-Dateien wird bei Erreichen des Datenträgerendes zum Datenträgerwechsel aufgefordert. Der alte Datenträger wird automatisch mit einem *Pseudo-CLOSE* abgeschlossen, der neue automatisch mit einem *Pseudo-OPEN* eröffnet.

Bei Mehrlaufwerksbetrieb (die Datei erstreckt sich über mehrere Disketten, die sich gleichzeitig in verschiedenen Laufwerken befinden) ist ein Datenträgerwechsel nicht erlaubt.

b) Direkt organisierte Datei

Die Datensätze sind entsprechend ihrer Satzadresse (Folge natürlicher Zahlen relativ zu 0) in aufsteigender Reihenfolge auf dem Datenträger abgelegt. Bei einem Zugriff zu einem beliebigen Satz

Tab. 2 Einteilung der Indexspur

Sektornummer	Verwendung
1–4 (oder 1	Systemlader (ab Version 0.8, vorher nur reserviert) Nachlader)
5	Fehlerkennsatz ERMAP
6	reserviert
7	Datenträgerkennsatz VOL1
8–26	Dateikennsätze HDR1 (jeweils ein Dateikennsatz pro Datei)

Position	Verwendung
1–4	Kennsatzidentifikator HDR1
5	reserviert
6–22	Dateiname (bei SIOS auf 8 Zeichen eingeschränkt)
23–27	Blocklänge
28	Satzmerkmal
29–33	Anfang der Datei (BOE)
34	Physische Satzlänge (gleich dem Wert in VOL1)
35–39	Ende der Datei (EOE)
40	reserviert
41	Übergehungsanzeiger
42	Dateischutzinformation
43	Schreibschutz (R/W, R/O)
44	Datenaustauschniveau
45	Mehrdatenträgeranzeige (Multi-Volume)
46–47	Datenträgerfolgenummer
48–53	Erstellungsdatum
54–57	Logische Satzlänge
58–62	nächster freier Satzraum
63–66	Anzahl unsortierter Zugänge bei schlüsselindizierter Datei
67–72	Verfallsdatum
73	Prüfungs- und Kopieranzeiger
74	Dateiorganisation (sequentiell, nicht sequentiell)
75–79	Ende der Daten (EOD)
80	reserviert

Tab. 3 Aufbau der Dateikennsätze bei SIOS

der Datei berechnet das System automatisch aus Satznummer und Satzlänge Spur- und Sektorposition auf der Diskette und führt diesen Zugriff dort direkt aus. Die Satzadresse bleibt dabei erhalten, auch beim Schreiben. Beim Zugriff oberhalb EOD (aber kleiner EOE) wird EOD auf den Wert *Satzadresse + 1* gestellt. Multi-Volume-Betrieb ist nur im Zusammenhang mit Mehrlaufwerksbetrieb möglich, das heißt nur auf allen im direkten Zugriff befindlichen Disketten. Dabei werden aus der Satzadresse vom System automatisch das Laufwerk sowie Spur- und Sektorposition berechnet.

c) Datei mit schlüsselindiziertem Zugriff (indexsequentiell)

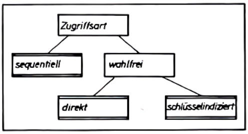
Eine Datei, für die schlüsselindizierter Zugriff spezifiziert ist, wird in Segmente eingeteilt, wobei der Index eines jeden Segments gleich dem Schlüssel des letzten Satzes im Segment ist. Als Schlüsselbegriffe werden alphanumerische Begriffe eingesetzt, die nicht in eine Folge von Satznummern überführt werden können (z. B. Name, Kontonummer o. ä.). Der Schlüsselbegriff muß aber im Datensatz enthalten sein. Die Sätze der Datei sind nun entsprechend den binären Werten ihrer Schlüssel zu ordnen (z. B. mit Dienstprogrammen SORD, MIXD aus sequentiell erfaßter Datei in schlüsselindizierte umsortieren).

Zur Realisierung des schlüsselindizierten Zugriffs wird eine Indextabelle im RAM benötigt, die zu jedem Segment den Schlüssel ausweist. Sie wird bei SIOS mit dem ersten Lesebefehl (READK) erzeugt. Bei jedem weiteren READK wird zunächst die Indextabelle durchsucht, bis der Schlüssel kleiner oder gleich dem Index i ist. Anschließend wird das Segment i sequentiell weiter nach dem spezifizierten Satz durchsucht. Mit dieser Methode werden alle Suchoperationen in großen Datenbeständen wesentlich beschleunigt. Multi-Volume-Betrieb ist ebenso wie bei direkter Dateiorganisation (siehe b) nur als Mehrlaufwerksbetrieb realisierbar.

3.3.3. Kassettendateien

Die Magnetbandkassette läßt sich effektiv nur als sequentiell organisierter Datenträger einsetzen. Bei SIOS ist der Einsatz sowohl für Programme als auch für Daten möglich. Die gesamte Software zur Programmentwicklung läuft allerdings nur auf Disketten.

Abb. 2 Zugriffsarten bei SIOS-Diskettendateien



Die physische Dateiaufzeichnung erfolgt blockweise mit Blocklängen zwischen 2 und 256 Byte. Aus zeitlichen Gründen sollte die Blocklänge jedoch mindestens 16 Byte betragen. Für Markierungszwecke spielt ein spezieller Block der Länge 2, die Bandmarke, eine besondere Rolle. Es ist möglich, beide Spuren der Kassette (Seite A und B) für die Datenaufzeichnung zu verwenden, sie werden jedoch als selbständige physische Einheiten (VOLUME) betrachtet. Für die logische Einteilung lehnt sich SIOS wie bei der Diskette an internationale Normen an, um den Datenaustausch zu gewährleisten. Neben der ISO TC 97 Sc 11 Nr. 35 findet die ECMA 41 ihren Niederschlag in der KROS 5109. In diesen Standards sind die drei Stufen der Dateiorganisation beschrieben: BASIC, COMPACT und EXTENTED. Davon werden bei SIOS die ersten beiden realisiert. EXTENTED entspricht der von den Großrechnern her bekannten Organisationsform auf $^1/_2$"-Magnetband und wird auf Grund des dafür erforderlichen hohen Aufwandes in SIOS nicht unterstützt. Zusätzlich wird noch die Stufe SIMPLE eingeführt, mit der über das Anwenderprogramm jede beliebige andere Dateiorganisation nachbildbar ist. Die realisierten Formen sind in Tab. 4 schematisch dargestellt.

- **SIMPLE** gestattet formatfreies Arbeiten. Je Kassettenseite ist nur eine Datei möglich, die durch eine Bandmarke abgeschlossen wird. Eine zweite Bandmarke markiert das Ende der Daten (EOD) auf der Spur.
- **BASIC** ist die unterste Stufe der Norm. Auch hier wird ohne Kennsätze gearbeitet. Es kann gleichfalls nur eine Datei je Kassettenseite aufgezeichnet werden. Die Datei ist in Bandmarken einzuschließen. Das Ende der Daten wird wiederum mittels Bandmarke markiert.
- **COMPACT** ist die zweite Stufe der Norm und arbeitet mit Kennsätzen. Die Kennsätze bestehen aus Blöcken der Länge von 32 Byte. Von den in der Norm möglichen vier Kennsätzen werden in SIOS zwei unterstützt:

– Dateianfangskennsatz (HDR)
– Dateiendekennsatz (EOF)

Dadurch ergibt sich die Möglichkeit zur Aufzeichnung mehrerer Dateien auf einer Spur. Die Kennsätze müssen aber außerhalb des Anwenderprogrammes mit einem speziellen Dienstprogramm (CATM) erzeugt werden.

3.4. Arbeitsspeicheraufteilung

Ausgehend von der Tatsache, daß für den Arbeitsspeicher sowohl RAM- als auch ROM-Bausteine Verwendung finden, muß die Einordnung des Betriebssystems im Adreßraum des Arbeitsspeichers sehr flexibel gestaltet sein. Es sind folgende Prämissen zu beachten:

– Der Bildwiederholspeicher für den Monitor besteht aus RAM und befindet sich adreßmäßig immer am Ende des RAM.

– Das Betriebssystem benötigt einen RAM-Speicherbereich als Verständigungsbereich (für Systemdaten). Dieser Bereich ist 2 KByte groß (bei der Speichererweiterung auf 112 KByte beträgt seine Größe 3 KByte) und liegt am Anfang des Adreßraumes (von 0 bis 7FFH bzw. OBFFH).

– RAM ist grundsätzlich vor ROM im Adreßraum einzuordnen. Sowohl RAM als auch ROM müssen in sich zusammenhängend sein.

– Nach dem ROM-Bereich darf bis zum Schluß des Adreßraumes eine Lücke sein.

– Anwenderprogramme können sich prinzipiell im RAM und auch im ROM (Festprogramm) befinden. Sie stehen aber immer hinter den Betriebssystemmoduln.

Abb. 3 zeigt eine gemischte RAM/ROM-Variante, die diesen Prämissen genügt. Derartige Arbeitsspeicherkonfigurationen und reine ROM-Varianten haben aber zwischenzeitlich an Bedeutung verloren. Deshalb ist in Abb. 4 noch eine typische RAM-Variante angegeben.

In diesem Zusammenhang ist es notwendig, gesondert auf die speichererweiterte Konfiguration einzugehen. Der Arbeitsspeicher besteht hier aus zwei Speicherseiten zu 64 KByte und 48 KByte RAM. Die Steckeinheit mit 48 KByte dient immer zur Speicherung chers und Auswertelogik auf dieser Steckeinheit).

Gerät	logische Geräteadresse	physische Geräteadresse
Dialoggerät	00	D4H
Tastatur	01	OEH
DFUE-Empfangskanal	02	24H
DFUE-Sendekanal	03	3AH
FD-Laufwerk 1	04	50H
FD-Laufwerk 2	05	66H
FD-Laufwerk 3	06	7CH
FD-Laufwerk 4	07	92H
1/2"-Magnetbandgerät	08	A8H
SLE oder HLE 3	09	BEH
Bildschirm	OAH	D4H
Drucker oder IFSS 3	OBH	EAH
Zweitdrucker oder IFSS 2	OCH	00
Kassettenmagnetbandgerät 1	ODH	7CH
Kassettenmagnetbandgerät 2	OEH	92H
LBE K 6200	OFH	16H
oder HLE 4	OFH	42H
IFSS 1	10H	2CH
V.24	11H	42H
HLE 1	12H	16H
HLE 2	13H	2CH
Programmein-, -ausgabegerät	15H	50H

Tab. 4 Logic Unit Block (Standardzuweisungen)

von SIOS. Sie enthält neben den Speicherbausteinen noch eine zusätzliche Logik, die die Zugriffe auf die andere Speicherseite steuert. Auf der 64-KByte-Seite befinden sich nur noch der Bildwiederholspeicher und der Verständigungsbereich, so daß der verfügbare Anwenderbereich hier eine Größe von 59 KByte erreicht. Da das Betriebssystem normalerweise kleiner als 48 KByte ist, bleibt auf der Betriebssystemseite immer ein gewisser RAM-Bereich frei. Dieser ist aber vom Anwender nicht belegbar. Die schematische Darstellung einer Arbeitsspeicheraufteilung zeigt Abb. 5.

Für alle Befehle, die zum Arbeitsspeicher zugreifen, gilt folgendes Arbeitsregime:

1. Zugriffe erfolgen immer auf die gleiche Seite, wo sich auch der Zugriffsbefehl selbst befindet.
2. Wenn vom Anwenderprogramm zum Betriebssystem zugegriffen werden muß, dann erfolgt die Seitenumschaltung grundsätzlich über Betriebssystem-Rufe (besondere Gruppe von Befehlen).
3. Wenn vom Betriebssystem auf die Anwenderseite zugegriffen werden muß, dann wird das über eine spezielle Hardware ausgelöst (neuntes Bit für jede Speicherzelle des 48-KByte-Spei-

chers und Auswertelogik auf dieser Steckeinheit).

Da die Speicherseitenumschaltung nicht wie sonst üblich über Software (OUT-Befehl) sondern über Hardware realisiert wird, erfolgt die Umschaltung praktisch innerhalb der normalen Befehlsausführungszeit. Sie wird deshalb vom Anwender überhaupt nicht wahrgenommen. Für ihn entsteht der Eindruck, daß er an einer ganz normalen Anlage arbeitet, die aber einen wesentlich größeren Anwenderspeicher besitzt.

Diese Lösung verlangt eine besondere Betriebssystemversion (Versionsnummer > = 2.0), was beim Generieren mit dem Dienstprogramm SGEN zu beachten ist.

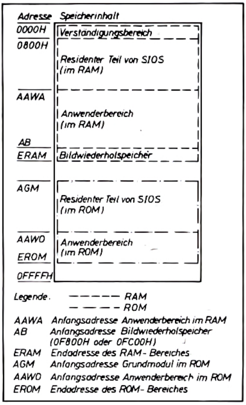
Es muß noch erwähnt werden, daß diese Lösung nur für das Betriebssystem SIOS verfügbar ist.

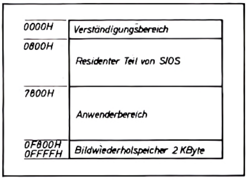
3.5. Kommando- und Dienstsystem

3.5.1. Der Monitor

Aufruf des Monitors

Der Monitor ist Bestandteil des residenten Teils von SIOS, des Betriebssystem-





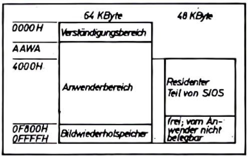


Abb. 3 Speicheraufteilung einer gemischten RAM/ROM-Variante

Abb. 4 Speicheraufteilung einer typischen RAM-Variante als Beispiel

Abb. 5 Speicheraufteilung bei der speichererweiterten Variante auf 112 KByte

kerns. Der Aufruf erfolgt, wie bereits unter 3.2. beschrieben, nach der Systeminitialisierung. Aber auch beim Beenden eines Programmes, nach bestimmten Fehlerquittungen und durch das Betätigen der Monitortaste wird die Programmroutine Monitor im System angesprungen.

Der Monitor dient der Kommunikation des Bedieners mit dem Betriebssystem. Dies erfolgt mittels Tastatureingaben durch den Bediener und Meldungen des Systems auf der untersten Zeile des Bildschirms, Systemzeile genannt.

Im Grundzustand hat die Systemzeile folgenden Aufbau:

MONITOR ==== SIOS 1526/x.y (M)

x.y gibt die Versionsnummer an.

Wurde das laufende Programm durch Betätigen der Monitortaste unterbrochen, so ist auf der Systemzeile folgende Anzeige zu sehen:

MONITOR ==== SIOS 1526/x.y (P)

Der Monitor ist in beiden Fällen bereit, Kommandos entgegenzunehmen und abzuarbeiten.

Die einzelnen Monitorkommandos werden durch jeweils drei Buchstaben gekennzeichnet. Ihre Eingabe ist mit ET1 (oder „ENTER“) abzuschließen.

Mit ET2 (oder „CNCL“) wird jedes Kommando beendet oder abgebrochen. Bei fehlerhaften Eingaben des Bedieners wird zur letzten Eingabe zurückgekehrt.

Im folgenden werden die wichtigsten Monitorkommandos erläutert.

Gerätezuweisung

Jedem logischen Gerät werden sowohl eine logische als auch eine physische Geräteadresse zugewiesen. Die logische Geräteadresse wird z.B. in den Befehlen der Makrobefehlssprache MABS genutzt. In der Tab. 4 ist die Übersicht für alle standardmäßigen Gerätezuweisungen enthalten. Sie entspricht dem Inhalt des Logik Unit Block (LUB) nach der Systeminitialisierung. Auf den LUB greifen verschiedene Systemkomponenten zu. Er belegt die Adressen 0300H ... 0315H im Speicher. Der L-Teil der jeweiligen Adressen ist die logische, der betreffende Speicherinhalt die physische Geräteadresse. Sind der logische oder der physische Gerätename doppelt belegt (z. B. SLE und HLE 3, Drucker und IFSS 3, FD-Laufwerk 3 und Kassettenmagnetbandgerät 1), so schließen sich diese peripheren Geräte oder Anschlüsse in einer Anlage gegenseitig aus.

Mit dem Kommando ASN kann die der logischen Geräteadresse zugeordnete physische Geräteadresse angezeigt und auch geändert werden.

Wichtig ist dieses Kommando vor allem für die Änderung des Programmein-/ausgabegerätes. Diesem logischen Gerät kann eines der physischen FD-Laufwerke oder Kassettenmagnetbandgeräte zugeordnet werden. Standard ist das erste FD-Laufwerk.

Speicherinhalt anzeigen oder ändern

Das Kommando wird mit MEM aufgerufen. Anschließend ist die Adresse der gewünschten Speicherzelle einzugeben. Eingaben und Ausgaben erfolgen jeweils in hexadezimaler Schreibweise.

Vor der Anzeige des nächsten Speicherinhaltes mit ET1 kann auch der Wert des aktuellen Bytes geändert werden. Änderungen innerhalb des Systembereichs werden nicht ausgeführt.

Ausgabe auf Diskette oder Kassette

Dieses Kommando ermöglicht die Aufzeichnung von Speicherbereichen auf Diskette oder Kassette. Es wird mit POU aufgerufen.

Ob Diskette oder Kassette und welches Laufwerk, das ist mit der logischen Geräteadresse 15H – dem Programmein-/ausgabegerät – festgelegt.

Bei einer Diskettenaufzeichnung sind Spur und Sektor, bei der die Aufzeichnung beginnen soll, und Sektoranzahl vom Bediener einzugeben. Bei Kassettenaufzeichnung sind die Angaben Anzahl der zu überlaufenden Blöcke, Anzahl der aufzuzeichnenden Blöcke und die Angabe der Seite (A oder B) erforderlich.

Durch eine weitere Bedienereingabe ist die Adresse des Hauptspeicherbereichs vorzugeben. Wird sie weggelassen, so wird der Speicherbereich ab AAWA auf Diskette oder Kassette aufgezeichnet.

Eingabe von Diskette oder Kassette in den Speicher

Mit dem Kommando PLO läßt sich der Inhalt von Disketten oder Kassetten in

den Speicher laden. Die weiteren erforderlichen Angaben sind die gleichen wie bei POU.

Programmaufruf
Das ist das wichtigste Monitorkommando, es wird mit CAL aufgerufen. Anschließend ist der Name des Programms einzugeben, der maximal fünf Zeichen lang sein darf. Das Programm muß sich unter dieser Bezeichnung als Phase in der Phasenbibliothek des Programmein-/ausgabegerätes befinden (siehe 3.7.).
Der Programmstart erfolgt bei AAWA oder einer in der Phase definierten Startadresse.

Programmladen
Durch die Eingabe von LOD und einem Programmnamen wird wie bei CAL die benannte Phase von einer Diskette oder Kassette aufgerufen und als Programm in den Speicher geladen. Im Gegensatz zu CAL wird das Programm aber nicht gestartet.
Die Ladeadresse kann mittels zusätzlicher Eingabe nach dem Programmnamen vom Bediener vorgegeben werden, sonst ist AAWA die Anfangsadresse.

Programmstart
Ein im Hauptspeicher befindliches Programm wird mit dem Monitorkommando NEW oder RUN gestartet. Während mit RUN nur Programme gestartet werden können, die auf der Adresse AAWA beginnen, ist es bei NEW möglich, eine beliebige Startadresse einzugeben.

Programmfortsetzung
Wurde während der Programmabarbeitung die Monitortaste betätigt, so wird der Programmablauf gestoppt und der Monitor aufgerufen. Soll dann wieder das unterbrochene Programm fortgesetzt werden, so ist das Monitorkommando RUN einzugeben.

Definierter Programmabbruch
Soll ein Anwenderprogramm definiert abgebrochen werden (z. B. nach nicht behebbaren Fehlern), ist mit der Monitortaste das Programm zu unterbrechen und anschließend dieses mit CAN abzubrechen.

Rückspulen der Kassette
Mit REW und der Angabe der logischen Geräteadresse des gewünschten Laufwerkes (ODH oder OEH) wird die eingelegte Kassette bis zum Klarsichtband zurückgespult.

Vorspulen der Kassette
Mit FOR und der Angabe der logischen Geräteadresse wird die eingelegte Kassette bis zum Klarsichtband vorgespult.

Systemzeit anzeigen und ändern
Hierfür ist das Monitorkommando CLK zu nutzen. Wird es aufgerufen, so erscheint die Systemzeit im Format hh.mm.ss in der Systemzeile (hh ... Stunden, mm ... Minuten, ss ... Sekunden). Die Systemzeit wird mit der Systeminitialisierung gestartet. Betätigt der Bediener die ET1-Taste, so ist anschließend die neue Zeit im Format hhmmss einzugeben.

Nachladen
Das Monitorkommando SYC bewirkt das Nachladen von Betriebsmoduln (s. 3.2.).

Ausschalten des Bürocomputers
Vom Monitorzustand aus kann mit OFF der Computer ausgeschaltet werden.

3.5.2. Dienst- und Hilfsprogramme

Die Zahl der zur Verfügung stehenden Dienst- und Hilfsprogramme ist sehr groß. Dies ist vor allem auf die Vielfalt der möglichen peripheren Anschlüsse und der mit dem Bürocomputer zu lösenden Aufgaben zurückzuführen. In diesem Abschnitt sollen die wichtigsten Dienst- und Hilfsprogramme für die Arbeit mit der Diskette und der Kassette, aber im Überblick auch für die anderen peripheren Geräte oder Anschlüsse vorgestellt werden.

INIT
Auf die Bedeutung dieses Programmes wurde bereits hingewiesen (s. 3.3.1.). Jede neue Diskette ist zu initialisieren. Aber auch Disketten, die aus den verschiedensten Gründen in einen Grundzustand gebracht werden sollen, sind zu initialisieren.
Beim Initialisierungsvorgang werden physische Sektoren der Länge 128, 256, 512 oder 1024 Byte in einer vom Bediener auszuwählenden Sektorfolge aufgezeichnet.
Die Sektorfolge gibt an, wie die Sektoren 1 – n auf der Spur verteilt sind.
Die Spur 0 (Indexspur) enthält stets 26 Sektoren in natürlicher Sektorfolge (auf Sektor n folgt Sektor n+1, auf Sektor 26 der Sektor 1) mit je 128 Byte. Es werden der Datenträgerkennsatz definiert und der Bereich der Dateikennsätze vorbereitet.
Außerdem werden die maximal zugelassenen zwei fehlerhaften Spuren (nicht Spur 0) in den Fehlerkennsatz eingetragen.
Nachdem alle Spuren initialisiert sind, erfolgt ein Kontrollesen aller Spuren (außer den möglichen zwei fehlerhaften Spuren). Werden hierbei Fehler festgestellt, wird bis zu zweimal der Initialisierungsvorgang wiederholt. Sollte es dann noch immer zu Fehlern kommen, so werden diese auf dem Bildschirm gemeldet.
Das Initialisieren erfolgt mit der Diskette im Laufwerk mit der logischen Geräteadresse 05H.

SGEN
Dieses Programm kann drei Aufgaben erfüllen:
– Generieren eines Systemladers für Systemdiskette oder Systemkassette für die RAM-Variante
– Generieren eines Nachladers auf einer Diskette oder Kassette
– Kopieren des Programms SGEN und der Datei SYSORGxx von Diskette auf Kassette.

Die Datei SYSORGxx (xx ... Versionnummer) enthält alle Betriebssystemmoduln der Version xx. Die SYSORGxx-Datei ist bei 5,25"-Disketten eine Multi-Volume-Datei.
Für die ersten beiden Aufgaben können vom Bediener über Bildschirmdialog die Moduln entsprechend der vorliegenden Gerätekonfiguration ausgewählt werden. Daraufhin werden gemäß den Angaben zur Systeminitialisierung (s. 3.2.) der Systemlader und die Betriebssystemdatei bzw. der Nachlader und das Nachladesystem auf der Diskette oder Kassette aufgezeichnet. Die spezifizierte Moduldatei heißt bei der Diskette SYSAARExx.
Das Programm SGEN ermöglicht es auch, ein Druckprotokoll über das erstellte System anzufertigen.

SOPH
Dieses Programm ermöglicht den Ausdruck des Inhalts der SYSORGxx-Datei einer Diskette. Es werden die Bezeichnung der einzelnen Betriebssystemmoduln, ihre Lage und Größe, ihre EDC-Zeichen und die Größe der gesamten Datei (Angabe von BOE und EOE) ausgegeben. Alle Betriebssystemmoduln bestehen aus 1- n Segmenten zu je 1 KByte. Für jedes Segment werden für Kontrollzwecke (z. B. beim Systemladen) EDC-Zeichen gebildet. Sie belegen die letzten beiden Byte des Segments.

SYSD
Dieses Dienstprogramm ermittelt den Aufbau der Betriebssystemdatei (SYSAREXX) oder des Nachladesystems einer Systemdiskette. Es werden deren Größe und Zusammensetzung auf den Bildschirm und wahlweise auch auf den Drucker ausgegeben.

FGEN
Zur Realisierung der Dateiarbeit mit Disketten ist FGEN ein unerläßliches Hilfsmittel. Mit ihm können die Dateikennsätze (HDR1) eingerichtet, angezeigt, geändert und gelöscht werden. Ebenso kann vom Bediener der freie Bereich auf der Diskette mit Spur- und Sektorangabe abgefragt werden.
Das Programm ermöglicht auch die Anzeige und Änderung des Datenträgerkennsatzes (VOL1). Dieser wird beim Initialisieren zum ersten Mal (mit Datenträgername: ROBINI) definiert.

DINH
Mit diesem Dienstprogramm läßt sich der Disketteninhalt sektorweise anzeigen oder ausdrucken. Die Sektorgröße kann 128 Byte oder 256 Byte sein. Die Zeichen werden hexadezimal oder alphanumerisch ausgegeben. Als Eingangsdaten sind Spur und Sektor vom Bediener vorzugeben.

DIDI
Hiermit kann der Inhalt geblockter und ungeblockter Dateien satzweise (entsprechend der logischen Satzlänge der Datei) oder sektorweise (entsprechend der physischen Sektorlänge der Diskette) auf dem Bildschirm angezeigt werden. Die Anzeige ist hexadezimal und/oder alphanumerisch.
Multi-Volt-Dateien sind zugelassen, es müssen sich aber alle Disketten dieser Datei in Zugriff – also in den einzelnen Laufwerken – befinden.

CODP
Zum physischen Kopieren von Disketten ist dieses Programm vorgesehen. Beide Disketten, die Quell- und die Zieldiskette, müssen gleiche Sektorlänge, die Spur 0 stets das Format 128 Byte/Sektor haben.
Da die Angabe der zu kopierenden Spuren möglich ist, können auch Daten von 8"-Disketten auf 5,25"-Disketten und umgekehrt kopiert werden. Standardannahme ist aber das Kopieren der gesamten Diskette (Spur 0-74, bzw. 0-37).
Auftretende Fehler werden protokolliert, über die Fortsetzung bei Fehlern kann der Bediener vor dem Kopiervorgang entscheiden (z. B. fehlerhafte Sektoren mit 00 oder 0FFH auf der Zieldiskette füllen).

COPD
Das Dienstprogramm COPD dient dem logischen Kopieren von Diskettendateien. Je nach Bedienereingaben sind folgende Funktionen möglich:
– Kopieren einer Datei, wobei der Datei auf der Zieldiskette ein neuer Name gegeben werden kann
– Kopieren einer Datei an das Ende (EOD) einer auf der Zieldiskette vorhandenen Datei
– Kopieren aller Dateien
– Kopieren eines Teils einer Datei.
Dieser Teil wird durch einen Beginnschlüssel und durch einen Endeschlüssel definiert. Die Anfangsposition des Schlüssels im Satz ist vom Bediener vorzugeben, außerdem der Beginn- und Endeschlüssel selbst. Sie können maximal 50 Positionen einnehmen. Positionen, die nicht bewertet werden sollen, sind mit dem Wert 00 anzugeben.
Der Teil kann als eine neue Datei aufgezeichnet werden. Es besteht aber auch die Möglichkeit, diesen Teil der Quelldatei an eine auf der Zieldiskette bestehende Datei anzuhängen. Als Endeschlüssel ist mit E auch der letzte Satz der Datei angebbar.
Beim Kopieren werden im Falle neuentstehender Dateien diese auf der Zieldiskette reorganisiert. Markierte Sätze können je nach Bedienereingabe kopiert oder nicht kopiert werden.

SORD
Dies ist ein Sortierprogramm für Diskettendateien. Aus einer unsortierten Datei wird eine neue sortierte Datei erzeugt, wobei die physische Satzlänge beliebig ist.
Das Sortiermerkmal kann aus maximal 32 Stellen bestehen und wird durch seine relative Satzposition vorgegeben. Der Vergleich erfolgt dezimal oder binär.
Die Sortierfolge kann aufsteigend oder absteigend sein.
Die sortierte Datei muß vor dem Aufruf von SORD mittels FGEN angelegt worden sein. Sie muß den gleichen Dateiaufbau wie die unsortierte Datei besitzen (Blockung, Blocklänge, logische und physische Satzlänge).

MIXD
Mit diesem Dienstprogramm können zwei sequentielle Quelldateien, die bereits sortiert sein müssen, zu einer neuen sortierten sequentiellen Datei zusammengefaßt werden.
Wiederum ist die Zieldatei vorher mit FGEN anzulegen. Als Sortiermerkmal sind auch maximal 32 Positionen innerhalb des Satzes zugelassen. Die Sortierung kann zwischen ab- und aufsteigend gewählt werden.
Der Ablauf des Programms kann auf der letzten Bildschirmzeile verfolgt werden.

PRDI
Soll eine geblockte oder ungeblockte Datei ausgedruckt werden, so ist das mit diesem Dienstprogramm möglich.
Je nach Bedienereingabe können ein, mehrere oder alle Sätze einer Datei auf den Drucker ausgegeben werden. Das Ausgabeformat – hexadezimal oder alphanumerisch – läßt sich auch angeben. Vor jeder Ausgabe des oder der Sätze wird der Dateikennsatz ausgedruckt.

KONV
Mit diesem Dienstprogramm wird aus einer aufsteigend sortierten sequentiellen Datei eine gestreute Datei erstellt. Der Schlüsselbegriff darf in der Quelldatei nicht doppelt vorkommen. Er muß rein numerisch und darf nicht länger als

50 Stellen groß sein. Der erste Schlüsselbegriff legt den ersten Satz der gestreuten Datei fest. Alle weiteren Sätze werden entsprechend ihres Schlüsselbegriffs eingeordnet, so daß aus der Differenz (+1) eines beliebigen mit dem ersten Schlüssel die Satznummer in der gestreuten Datei festgelegt ist und ein schneller Direktzugriff auf den beliebigen Satz vorbereitet werden kann. Ein aufwendigeres sequentielles Suchen eines Datensatzes kann so umgangen werden. Das bedingt aber, daß bei KONV Lücken im Schlüsselsystem der sequentiellen Datei zu als gelöscht markierten Datensätzen in der gestreuten Datei führen. Ein relativ geschlossenes Schlüsselsystem in der sequentiellen Datei ist somit eine Voraussetzung zur sinnvollen Anwendung von KONV.

CODK

Mittels CODK können sowohl Diskettendateien auf Kassette als auch Kassettendateien auf Diskette kopiert werden. Das Dateiformat der Kassette kann COMPACT, BASIC oder SIMPLE sein. Bei COMPACT wird der Dateiname von der Quelldatei für die Zieldatei übernommen. Wird von der Kassette eine Datei im BASIC- oder SIMPLE-Format kopiert, so ist der Dateiname für die Diskettendatei vom Bediener einzugeben. Während sich geblockte und ungeblockte Diskettendateien auf Kassette kopieren lassen, werden beim Kopieren von der Kassette nur ungeblockte Diskettendateien erzeugt.

COCA

Mit diesem Dienstprogramm lassen sich Kassettendateien aller Formate auf eine andere Kassette kopieren. Das Format der Zieldatei kann beliebig gewählt werden, also vom Format der Quelldatei abweichen. Es ist aber auch mit diesem Programm möglich, daß eine bereits existierende Zieldatei um die Quelldatei erweitert wird.

CATM

Um Kassetten für die Datenaufzeichnung vorzubereiten, ist das Programm CATM zu nutzen. Soll mit dem SIMPLE- oder im BASIC-Format gearbeitet werden, so wird durch CATM eine Bandmarke (Endemarke) am Bandanfang aufgezeichnet. Für das COMPACT-Format wird entweder am Anfang des Bandes (bei leerer oder zu überschreibender Kassette) oder am Datenende (es existieren bereits COMPACT-Dateien auf dem Band) eine COMPACT-Leerdatei erzeugt. Neben dem Dateinamen kann der Bediener auch eine Kassettennummer und das Erstellungsdatum für den Dateianfangskennsatz (HDR) eingeben. Eine weitere Funktion des Dienstprogramms ist die sequentielle Anzeige aller aufgezeichneten Daten (Blöcke oder Bandmarken). Dabei kann auch ein Vorsetzen um n Datenblöcke ausgelöst werden. Ein Block kann 2...256 Byte groß sein und entspricht einem logischen Satz.

COCP

Dieses Dienstprogramm ist für das physische Kopieren von Daten- und Programmkassetten vorgesehen. Es ist jedes Format von Kassettendateien möglich.

MACK

Mit dem Programm MACK können – neben der Anzeige von Dateisätzen auf dem Bildschirm – verschiedene Änderungen an Kassettendateien ausgeführt werden:

– Anzeige von Dateisätzen auf dem Bildschirm
– eine Datei um einen oder mehrere Sätze erweitern
– Kopieren von Sätzen, wobei in die Kopie auch Sätze eingefügt oder Sätze der Kopie verändert werden können
– die kopierten Sätze können an eine andere Datei angefügt werden oder eine neue Datei bilden
– eine Datei oder ein Teil einer Datei kann mit einer Bandmarke versehen werden.

Es sind alle drei Formate von Kassettendateien zugelassen.

DIKA

Sollen Kassettendateien satzweise angezeigt werden, so ist dieses Dienstprogramm aufzurufen. Das Anzeigeformat kann vom Bediener festgelegt werden und hexadezimal und/oder alphanumerisch sein. Bei COMPACT-Dateien wird der Datei-Kennsatz angezeigt, nachdem der Name der Datei eingegeben wurde. Es kann ein beliebiger Satz der Datei ausgewählt werden, danach ist u. a. eine fortlaufende Anzeige der folgenden oder vorhergehenden Sätze möglich.

PRKA

Hiermit lassen sich Kassettendateien aller drei Formate vollständig oder satzweise ausdrucken. Die Zeichen werden in hexadezimaler oder alphanumerischer Schreibweise ausgegeben.

Dienstprogramme für die Arbeit mit der Lochbandeinheit K 6200

Für die Anzeige und den Druck der Zeichen von Lochbändern verschiedenster Lochband-Codes – wie z. B. R-300-Code, Fernschreibcode, GOST-Code – gibt es zwei Dienstprogramme. Mit dem Dienstprogramm DPTA werden jeweils 255 Zeichen und mit DPTP ein Satz vom Lochband eingelesen. Der Code für die Satzmarke und für die Bandendeerkennung sind beim Programm DPTP vom Bediener vorzugeben. Die Paritätskontrolle (paarig oder unpaarig), die Prüfbitausblendung, das Lesen oder Überlesen von NUL und DEL können wahlweise eingeschaltet werden. Ein weiteres Dienstprogramm COTP dient dem physischen Kopieren von 5- oder 8-Kanal-Lochbändern mit beliebigem Code und beliebiger Parität.

Dienstprogramme für die Arbeit mit dem $^1/_2$"-Magnetbandgerät CM 5300

Um Dateien von Diskette oder Kassette auf das $^1/_2$"-Magnetband zu übertragen, gibt es drei Konvertierungsprogramme: KOE, KOS und KOK. Sie erzeugen Dateien in den Formaten für ESER-Rechner, SKR-Rechner und für das Kleinrechensystem (KRS)-Format.

Weitere Dienstprogramme (LABE, LABS, LABK) dienen der Initialisierung von $^1/_2$"-Magnetbändern mit Kennsätzen und Bandmarken vorgenannter Formate. Sind die Magnetbänder bereits initialisiert („gelabelt"), so werden die vorhandenen Kennsätze angezeigt.

Das Dienstprogramm TADP ermöglicht die Anzeige und den Ausdruck des Inhaltes eines $^1/_2$"-Magnetbandes oder die Ausgabe einer Statistik über den inhaltlichen Magnetbandaufbau.

Dienstprogramm für die SLE K 6501 und HLE K 6503

Das Programm DPCC ist für das Lesen, Schreiben und die Anzeige der Zeichen

einer Plastkarte mit Magnetstreifen vorgesehen. Dabei wird den unterschiedlichen Hardwarebedingungen der beiden Geräte Rechnung getragen: Die Schreib-Lese-Einheit (SLE) kann alle Spuren der Plastkarte lesen und aufzeichnen. Die Handleseeinheit (HLE) liest nur die Spur 2 einer Karte, ein Schreiben oder Ändern des Spurinhaltes kann hier nicht ausgeführt werden.

Dienstprogramme für die Datenfernübertragung

EMBS – Emulationsprogramm zur Nachbildung des Abonnentenpunktes EC 7925 für ESER-Rechner (BSC III-Prozedur)

EM62 – Emulationsprogramm zur Nachbildung des Abonnentenpunktes EC 8562 / EC 8564 (asynchrone Prozedur)

DD62 – Programm zur asynchronen Datenübertragung zwischen Bürocomputer und ESER-Rechner auf Basis der AP 62/64-Prozedur

MD62 – Programm zur Steuerung der Datenübertragung zwischen zwei Bürocomputern auf Basis der asynchronen Datenfernübertragungsprozedur AP 62/64

EMBT – Emulationsprogramm, das weitgehend die Lochkartenlese- und -stanzstation IBM 2780 simuliert: Stapelübertragung auf Basis der synchronen Übertragungsprozedur BSC I

3.6. Betriebssystemkern (E/A-System)

3.6.1. Überblick

Die Realisierung aller Programme, ob Dienst-, Hilfs- oder Anwenderprogramme, erfolgt bezüglich der E/A-Arbeit stets über den Betriebssystemkern, den residenten Teil des Betriebssystems. Kennzeichnend für die Struktur dieses Teils des Systems ist sein streng modularer Aufbau. Der Betriebssystemkern läßt sich in zwei funktionelle Ebenen teilen, in die logische Ebene (MINT 1520) und in die physische Ebene (SIEX 1526). In Abb. 6 sind diese Ebenen und ihre Komponenten schematisch dargestellt. Während in der logischen Ebene die logische Verarbeitung der Daten, die Vorbereitung der Ein- und Ausgabe und die Fehlerbehandlung erfolgen, werden in der physischen Ebene die verschiedenen gerätetechnisch bedingten E/A-Routinen und deren Organisation unter der Berücksichtigung der Simultanarbeit realisiert.

3.6.2. Die logische Ebene

Der Makrobefehlsinterpreter MINT 1520 dient der Abarbeitung der in der Makrosprache MABS 1520 geschriebenen Programme. Er beinhaltet neben dem Interpreter selbst die Routinen zur logischen E/A-Arbeit (Dateiarbeit) und die Moduln zur Verarbeitung der arithmetischen und logischen MABS-Befehle. Diese liefern für häufig benötigte Operationen optimale CPU-Befehlsfolgen.

Folgende wichtige *Eigenschaften* kennzeichnen die logische Ebene des Betriebssystemkerns:

- Einzelprogrammabarbeitung
- allgemeiner Aufbau der MABS-Befehle als Zweiadreßbefehle
- indirekte Adressierung möglich
- anwendereigene Unterbrechungsbehandlungsroutinen für DFÜ und Zeitgeberanforderungen
- umfassende Dateiorganisation für Disketten:
  - Öffnen und Schließen von Dateien
  - Kennsatzprüfung
  - Unterstützung der sequentiellen, direkten und schlüsselindizierten Zugriffsmethode
- Unterstützung der drei Dateiformate für Kassettenmagnetband
- programmgsteuertes Einschalten des Interpreters von der CPU-Befehlsebene aus (mit CPU-Befehl RST 28H)
- programmgesteuertes Ausschalten des Interpreters durch Aufruf der CPU-Befehlsebene (mit MABS-Befehl EOI oder CALC) oder des Monitors (mit MABS-Befehl EOP)
- auf Grund der Modularität des Systems einfache Erweiterung um neue Moduln und damit leichte Realisierung verschiedener Gerätekonfigurationen im System.

In drei Arithmetikmoduln werden die Verrechnung, der Vergleich und andere logische Operationen von numerischen Werten in drei verschiedenen Darstellungen ausgeführt:

- Zahlen innerhalb von Zeichenketten im ISO-7-Bit-Code
- Dualzahlen
- Zahlen im BCD-Format (gepackte Zahlen).

Die Konvertierungen zwischen den einzelnen Darstellungen von Zahlenwerten werden ebenfalls in diesen Moduln realisiert.

Für jedes logische E/A-Gerät gibt es einen logischen E/A-Modul, wobei für mehrere gleiche physische E/A-Geräte nur ein logischer Modul vorgesehen ist. So gibt es z. B. für die maximal im System zugelassenen vier FD-Laufwerke nur einen logischen Modul.

Zur Übergabe der Arbeiten von der logischen Ebene an die physische Ebene werden die physischen Rufe genutzt. In diesen Rufen werden Parameter definiert, die die physische Ebene zur Realisierung des physischen E/A-Prozesses benötigt. Der physische Ruf stellt somit die Schnittstelle zwischen den beiden Ebenen des E/A-Systems dar. In ihm können je nach Art des E/A-Prozesses folgende Informationen definiert werden:

- das Steuerbyte mit Informationen über den Ruf selbst
- die logische Geräteadresse
- das Kommandobyte zur Beschreibung der auszuführenden Operation (z. B. Aufzeichnen, Wiedergabe, Rückspulen, usw.)
- die Adresse für einen Rückgabecode
- die Adresse und die Länge eines Ein- oder Ausgabebereiches im Hauptspeicher
- die Angaben für die Positionierung des E/A-Gerätes (z. B. Spur und Sektor beim FD-Laufwerk).

Gekennzeichnet ist der physische Ruf durch den CPU-Befehl RST 08H an erster Position. Ihm folgt direkt das Steuerbyte. Dieses enthält unter anderem die Information, ob die anderen Parameter direkt folgen oder in einem RAM-Bereich (Verständigungsbereich des Betriebssystems) bereitgestellt werden. Der letztere Fall ist der häufigere, da die Parameter bei jedem E/A-Befehl unterschiedlich sind. Durch den CPU-

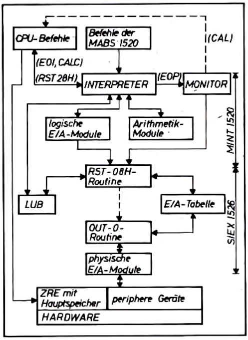


Abb. 6 Ebenen des Betriebssystemkerns

Befehl RST 08H wird die RST-08H-Routine aufgerufen, die der physischen Ebene des Systems zugeordnet ist. Aus Gründen der Modularität des Systems benutzt auch der Monitor für jede E/A-Aufgabe, wie zum Beispiel die Kommandoeingabe von der Tastatur oder die Ausgabe auf Diskette und Kassette mit POU, diese physischen Rufe. Die Parameter werden entsprechend der Kommandoeingabe eingestellt.

3.6.3. Die physische Ebene

Die physische Ebene mit der Bezeichnung SIEX 1526 beinhaltet die RST-08H-Routine, die OUT-0-Routine und die physischen Moduln. Sie ist eng mit der Hardware verbunden und realisiert den Betrieb der peripheren Geräte entsprechend der Vorgaben des physischen Rufes. In ihr werden alle Organisationsaufgaben zum gleichzeitigen Betrieb mehrerer peripherer Geräte übernommen, Kontrollfunktionen ausgeführt und alle hardwarespezifischen Funktionen zur Ansteuerung der Geräteanschlüsse durchgeführt.

Die Aufgaben der RST-08H-Routine

① Sie kontrolliert, ob das im physischen Ruf mit der logischen Geräteadresse festgelegte physische Gerät frei ist. Die Zuordnung des logischen Gerätes zum physischen Gerät ist mit dem LUB vorgegeben. Ist das betreffende Gerät besetzt, das heißt, es läuft noch ein E/A-Prozeß mit diesem Gerät, so wird in der Routine auf das Ende dieses aktiven Prozesses gewartet. Ein Warteschlangensystem wird nicht aufgebaut.

② Ist das Gerät frei, so werden die Parameter des Rufes in standardisierter Form in eine E/A-Tabelle übergeben und eine Kennung für die nunmehr anliegende Anforderung eines neuen E/A-Prozesses gesetzt.

Die E/A-Tabelle besteht aus 16 E/A-Worten. Diese Worte sind den einzelnen physischen Geräten zugeordnet, der L-Adreßteil des E/A-Wortes ist gleichzeitig die physische Geräteadresse. Jedes Wort ist 22 Byte groß. In ihm sind alle Informationen für den gesamten Prozeß einer E/A-Arbeit – von seinem Start bis zur kompletten Fertigstellung einschließlich Rückmeldung – enthalten. Das sind sowohl statische Steuerdaten, die mit Betriebsbeginn eingetragen werden und für die gesamte Arbeitszeit des Gerätes gültig bleiben, als auch dynamische Steuerdaten, die vom aktuellen Bearbeitungszustand des E/A-Prozesses abhängig sind. Zu den letzteren gehören zum Beispiel die vorgenannte Kennung zur Anforderung eines E/A-Prozesses und die vom physischen Ruf übernommenen Parameter. Die Zuordnung der logischen Geräteadresse zum E/A-Wort wird wieder dem LUB entnommen.

Das Vorhandensein von 16 E/A-Worten und nicht nur eines ist eine Voraussetzung zur Realisierung der Simultanarbeit.

③ Aufruf der E/A-Steuerung

Diese Aufgabe erfolgt mittels des CPU-Befehls OUT an die Toradresse 0 am Ende der RST-08H-Routine. Dadurch wird vom sogenannten Betriebssystem-PIO auf der ZRE K 2526 ein Interrupt ausgelöst, dessen Priorität innerhalb der Interruptkette des gesamten Gerätes die niedrigste ist. Die dazugehörige Interruptbedienroutine wird OUT-0-Routine genannt und ist Bestandteil der physischen Ebene des E/A-Systems.

Die Aufgaben der OUT-0-Routine

Die OUT-0-Routine arbeitet eng mit der E/A-Tabelle zusammen. Mit deren Informationen werden die Aufrufe der einzelnen physischen E/A-Moduln gesteuert, damit diese einen E/A-Prozeß vorbereiten, starten oder beenden. Dabei werden Zeitbedingungen, die BUS-Auslastung und andere Faktoren berücksichtigt. Weiterhin ermöglicht die Steuerroutine die Arbeit in Unterbrechungsprogrammebenen. Es sind drei Unterbrechungsprogramme möglich:

– für die Datenfernübertragung
– für den Zeitgeber
– für den Monitor.

Auslösende Wirkung dafür haben das Ende einer Datenfernübertragung, das Ende eines programmierbaren Zeitlimits bzw. das Betätigen der Monitortaste während einer Programmabarbeitung. Während letzteres den Monitor (mit Rückkehrmöglichkeit, s. 3.5.1.) aufruft, veranlaßt die OUT-0-Routine bei den ersten beiden den Aufruf einer Routine, die zu einer Mitteilung an die logische Ebene führt, in der es schließlich durch den Interpreter zum Start des entsprechenden Unterbrechungsprogrammes kommt. Das bedeutet, daß die Vorbereitung, Ausführung und das Beenden des Unterbrechungsprogramms durch die logische Ebene vorgegeben wird. Den Startpunkt der Ausführung jedoch steuert die physische Ebene auf Grund der genannten Ereignisse.

Die Aufgaben der physischen Moduln

Ein physischer Modul bedient die Anschlußsteuerung der peripheren Geräte; gleiche Geräte mit gemeinsamem Adapter (z. B. FD-Laufwerke) haben einen gemeinsamen physischen Modul. Die Moduln entnehmen der E/A-Tabelle alle nötigen Steuerdaten, sie benutzen aber auch Teile der Tabelle als Statusregister und Zwischenspeicher. Jeder physische Modul besteht aus drei Teilen.

Der *1. Teil* ist der physische Grundmodul. Er wird von der OUT-0-Routine als Unterprogramm aufgerufen und besitzt somit auch den Interruptstatus der OUT-0-Routine. Der Grundmodul eines physischen Moduls dient der Vorbereitung, dem Start und dem Beenden eines E/A-Prozesses. Außerdem erfolgt hier die Registrierung von Fehlern, um sie an die logische Ebene zu übergeben.

Der *2. Teil* ist der Datenübertragungsmodul. Er ist eine abgeschlossene Interruptbedienroutine, die durch die zuge-

ordnete Anschlußsteuerung aktiviert wird. Je nach räumlicher Anordnung des Adapters im Gerät und damit in der Interruptkette wird dieser Routine eine bestimmte Priorität zugeordnet. Mit der Interruptroutine erfolgt im allgemeinen die zeichenweise Datenübertragung vom oder zum peripheren Gerät. Ist das Ende eines E/A-Prozesses erreicht, werden keine Interupts mehr ausgelöst. Eine Endekennung zur Auswertung durch die OUT-0-Routine wird in der E/A-Tabelle gesetzt.

Der *3. Teil* des physischen Moduls ist die Betriebsbeginnroutine. Sie wird beim Start des Gesamtsystems aufgerufen und bringt die Anschlußsteuerung und in einigen Fällen das Gerät in einen definierten Grundzustand, zum Beispiel Löschen des Bildschirms. Es werden auch die statischen Steuerdaten in das zugehörige E/A-Wort eingetragen. Zum 3. Teil gehören somit auch solche Aufgaben wie das Setzen von Adreßzeigern, das Programmieren der E/A-Bausteine PIO, SIO und CTC, den LUB mit Standardwerten füllen, in die Interruptsäule die Adressen der Bedienroutinen eintragen.

3.6.4.
Die Simultanarbeit

Vom Betriebssystem wird die Simultanarbeit peripherer Geräte untereinander und zum Anwenderprogramm unterstützt. Ob diese Möglichkeit genutzt wird, hängt vom Anwenderprogramm ab. Alle E/A-Befehle der MABS 1520 können mit und ohne Warten auf das Ende des E/A-Prozesses programmiert werden. Soll die Simultanarbeit zum Zwecke der Programmlaufzeitverkürzung genutzt werden, so sind E/A-Befehle ohne Warten zu programmieren. Um die Ausführung des E/A-Befehls im weiteren Programmablauf testen zu können, sind dann die sogenannten WAIT-Befehle vorzusehen. Diese „warten" auf das Ende des E/A-Prozesses, wenn er noch nicht abgeschlossen war, und übergeben den Status über die Abarbeitung an das Programm. Kann auf diese Rückmeldung verzichtet werden, so ist auch ein WAIT-Befehl nicht erforderlich (außer am Programmende).

Die Realisierung und Überwachung der Simultanarbeit erfolgt im wesentlichen durch die OUT-0-Routine. Sie garantiert, daß jedem aktiven Gerät ausreichend Zeit zu seiner Bedienung zur Verfügung steht. Es werden von ihr nur so viel E/A-Prozesse zum Starten freigegeben, so daß es zu keiner gegenseitigen Störung auf dem Systembus kommt.

Hier soll kurz auf die Voraussetzung und auf die Bedingungen zur Realisierung der Simultanarbeit eingegangen werden:

Technische Voraussetzung für die Simultanarbeit ist ein komfortables Unterbrechungssystem, das durch das K-1520-System mit seiner INT- und BUSRQ/BUSAK-Steuerung gegeben ist. Weiterhin hat die Hardware zu gewährleisten, daß Geräte mit der kleinsten Antwortzeit die höchste Priorität in der Interruptkette erhalten. Die Antwortzeit ist die Zeit, in der das System eine Anforderung eines peripheren Gerätes oder Anschlusses unbedingt bedient haben muß, damit kein Datenverlust auftritt. Beträgt zum Beispiel die Übertragungsgeschwindigkeit eines Gerätes 10 kHz, so ist die Antwortzeit 0,1 ms. Somit haben periphere Geräte oder Anschlüsse mit hoher Datenrate eine kleine Antwortzeit.

Die Zeit, die das System benötigt, um eine Anforderung tatsächlich zu bedienen, ist die Bedienzeit. Sie entspricht im allgemeinen der Dauer der Interruptbedienroutine einschließlich der maximal möglichen Zeit des Interruptannahmezyklus.

Das Verhältnis von Bedienzeit zur Antwortzeit ist der Systembuszeitbedarf oder die Busbelastung und ergibt für jedes Gerät eine typische Größe. Geräte, die im Start-Stop-Betrieb arbeiten, deren Bedienung also beliebig lang unterbrochen werden kann, haben in diesem Sinne keinen Buszeitbedarf. Sie sind die zeitunkritischen Geräte (z. B. Drucker) und werden auch den Interruptebenen niederer Priorität zugeordnet. Die anderen Geräte sind die zeitkritischen (z. B. FD-Laufwerk).

Wenn alle Interruptbedienroutinen des ganzen Systems beliebig splittbar sind und die Summe der Busbelastung aller aktiven Geräte ständig kleiner 1 ist, so ist eine störungsfreie und zeitrichtige Unterbrechnung der einzelnen Routinen und damit die Simultanarbeit gewährleistet.

Die erste Forderung wurde bei der Programmierung des Systems beachtet. Von der OUT-0-Routine wird überwacht, daß die zweite in allen Betriebsfällen eingehalten wird.

Die zeitkritischen Geräte melden vom physischen Grundmodul aus ihren Bedarf beim Vorbereiten eines E/A-Prozesses an. Die OUT-0-Routine überprüft, ob die neue Gesamtbelastung kleiner 1 bleibt. Ist dies der Fall, wird der physische Grundmodul zum Starten des E/A-Prozesses aufgefordert. Ein E/A-Prozeß, der nicht sofort gestartet werden kann, wird dann freigegeben, wenn durch das Beenden eines anderen E/A-Prozesses die Gesamtbusbelastung genügend reduziert werden konnte. Die OUT-0-Routine kann nur wartende E/A-Prozesse durch den Aufruf des physischen Grundmoduls starten lassen. Darum hat der physische Grundmodul beim Beenden eines E/A-Prozesses der OUT-0-Routine dies und seine spezifische Busbelastung zu melden.

Für die Tastatur wird der benötigte Anteil an der Busbelastung ständig freigehalten. Die Interruptroutinen der zeitunkritischen Geräte laufen in den Zeitlücken ab, in denen keine Interruptroutinen der zeitkritischen Geräte aktiv sind.

3.7.
Programmentwicklungssoftware
3.7.1.
Vorbemerkungen

Um neue Programme zu erfassen und abarbeitungsfähig zu machen, ist ein mehrstufiger Prozeß erforderlich. Mit Hilfe verschiedener Verarbeitungs- und Dienstprogramme entstehen Zwischen- und Endergebnisse, die auf der Diskette in verschiedenen Dateien, hier Bibliotheken genannt, gespeichert werden.

Abb. 7 zeigt in einer schematischen Darstellung die wichtigsten Zusammenhänge bei der Programmentwicklung im Assemblersprachniveau. Die Mittel und Wege der Programmierung in den Spra-

chen BASIC, PASCAL und COBOL werden hier nicht dargestellt und erläutert.
Unter SIOS lassen sich zwei Assemblersprachen zur Programmentwicklung nutzen:
– die CPU-Befehlssprache des U 880, CABS 1520 (SIEX)
– die Makrobefehlssprache MABS 1520 (SIEX).
Die Programmierung von Ein- und Ausgaben zu den peripheren Geräten durch den Anwender ist nur mit den E/A-Befehlen der MABS 1520 zugelassen!
Um die Vorteile beider Befehlssprachen innerhalb eines Programms nutzen zu können, wird eine weitere Assemblersprache zur Verfügung gestellt: die SABS 1520 (SIEX). Diese umfaßt den gesamten Befehlsvorrat der Sprachen CABS und MBAS und die Anweisungen zum Umschalten zwischen diesen beiden, so daß ein gemischtes Programm vom Anwender geschrieben werden kann. Je nach Aufgabenstellung hat der Programmierer eine der Sprachen auszuwählen, an deren Befehlssatz, Pseudoanweisungen und sonstige Bedingungen er sich halten muß.
Disketten für die Programmentwicklung und -speicherung erfordern das physische Sektorformat von 128 oder 256 Byte/Sektor.

3.7.2. Bibliotheksservice

Für den Prozeß der Programmentwicklung sind die bereits genannten speziellen Dateien, die Programmbibliotheken, erforderlich. Nach ihrem Inhalt werden drei Bibliotheksformen auf der Diskette unterschieden:
- die Quelltextbibliothek, zum Speichern von Quelltextbüchern
- die Objektbibliothek, zum Speichern der Objektprogramme, auch Moduln genannt
- die Phasenbibliothek, zum Speichern der Phasen.

Die Phasen können adreßmäßig fest oder verschiebbar gestaltet sein und enthalten nicht nur den Programmcode, sondern auch Kennbytes, unter anderem zur Veränderung der Adressen dieses Codes. Sie sind vom Monitor oder vom MABS-Befehl CALP aufrufbar und werden dabei im Hauptspeicher als abarbeitbare Programme eingetragen.
Jede Bibliothek enthält ein Verzeichnis aller in ihr enthaltenen Bücher, Moduln bzw. Phasen. Jedes Buch, jeder Modul und jede Phase haben einen maximal fünfstelligen Namen.
Auf einer Diskette kann jede der drei Bibliotheken nur einmal vorhanden sein. Der physische Dateiname für die Quelltextbibliothek ist SOURCELB (von engl. source library), für die Objektprogrammbibliothek RELOCLB (von engl. relocable library) und für die Phasenbibliothek PHASELB (von engl. phase library).
Auf der Kassette kann nur eine Phasenbibliothek eingerichtet werden.
Mit dem Dienstprogramm LBSV wird der Anwender weitgehendst beim Aufbau und bei der Pflege dieser Bibliothek auf der Diskette unterstützt. Es stehen ihm durch das Programm folgende Möglichkeiten offen:
– Eröffnen, Erweitern und Verkleinern von Bibliotheken
– Löschen von Bibliotheken, Büchern, Moduln oder Phasen
– Druck von Bibliotheksverzeichnissen und Büchern
– Umbenennen von Büchern, Moduln und Phasen
– Kopieren von Bibliotheken, Büchern, Moduln und Phasen.
Alle diese Funktionen sind dialogorientiert gestaltet, das heißt, der Bediener wählt – durch ein Bildschirmmenü geführt – mittels Tastatureingaben die gewünschte Funktion aus. Auf den Bildschirm werden über die Ausführung der Funktion Informationen ausgegeben.
Die Anwendung des Dienstprogramms LBSV ist der erste Schritt der Programmentwicklung.

3.7.3. Erfassen des Quelltextes

Zum Erfassen ist das Programm EDIT 1520 (SIEX) zu nutzen. Dieser Editor ermöglicht, die verschiedenen Quelltexte der genannten Assemblersprachen in die Quelltextbibliothek einzugeben, den Quelltext eines Buches zu ändern, zu erweitern, zu löschen (teilweise oder

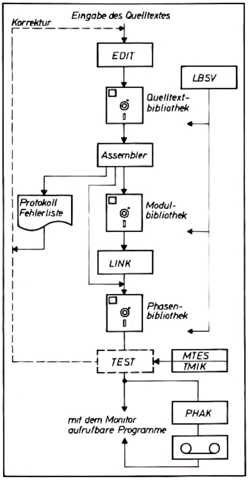


Abb. 7 Programmentwicklung im Assemblersprachniveau

völlig), anzuzeigen und zu drucken. Außerdem läßt sich mit dem Editor auch das Verzeichnis der Quelltextbibliothek anzeigen. Zur Lösung dieser Aufgaben ist im Editor ein Kommunikations- und Kommandosystem eingebaut, von dem aus die einzelnen dialogorientierten Funktionen aufgerufen werden können.
Die Quelltextbibliothek muß mittels LBSV auf der Diskette eingerichtet worden sein.

3.7.4. Übersetzen der Quelltexte

Um die mit dem Editor erfaßten Quelltexte zu übersetzen, sind die Assembler erforderlich. Je nach der verwendeten Assemblersprache, ist der entsprechende Assembler aufzurufen:

Assemblersprache	Assembler
CABS 1520	CASS 1520 (SIEX)
MABS 1520	MBAS 1520 (SIEX)
SABS 1520	SASS 1520 (SIEX)

Alle drei Assembler sind 2-Paß-Assembler und arbeiten sehr speicherintensiv. Wie beim Editor wird auch hier die Arbeit des Bedieners durch Bildschirmmenüs erleichtert. Die Hauptaufgabe der Assembler ist es, das vom Bediener vorzugebene Quelltextbuch entweder in einen Modul oder direkt in eine Phase zu übersetzen. Der Name des Moduls oder der Phase sind ebenfalls vom Bediener vorzugeben. Außerdem ist es möglich, Ausdrücke oder Anzeigen von Assemblerlisten, Symboltabellen und Fehlerprotokollen abzufordern. Bei den Assemblern CASS und SASS lassen sich auch mehrere Büeher in einem Aufruf angeben, die zu einem geschlossenen Modul oder einer geschlossenen Phase übersetzt werden.
Durch LBSV sind vorher die Modul- oder die Phasenbibliothek einzurichten.

3.7.5.
Programmverbinder

Die mit dem Assembler erstellten Moduln werden mit dem Programmverbinder LINK 1520 (SIEX) gebunden. Dabei entsteht eine Phase. Voraussetzung ist auch hier, daß durch LBSV eine Phasenbibliothek eingerichtet worden war.
Es können bis zu 250 Moduln zu einer Phase gebunden werden. Der Name der Phase und die Startadresse (hexadezimaler Wert oder ein externes Symbol) sind vom Bediener anzugeben. Für jeden Modul kann eine Basisadresse vorgegeben werden; wird diese Eingabe vom Bediener übergangen, so schließt sich der Modul direkt hinter den zuvor gebundenen Modul an.
Eine besondere Wirkung kann mit der aufrufbaren Funktion Autolink erzielt werden. Hierbei werden alle nicht auflösbaren externen Verbindungen als Modulname in allen zugängigen Modulbibliotheken (also in allen FD-Laufwerken) gesucht und gebunden.
LINK 1520 ermöglicht auch, Überlagerungsstrukturen zu erzeugen. Treten Fehler beim Bindevorgang auf (z. B. offene Verweise, Doppeldefinitionen), so werden diese wahlweise auf Bildschirm oder Drucker ausgegeben.

3.7.6.
Programmtest

Für das Testen der nunmehr entstandenen Phasen gibt es zwei Testprogramme: MTES 1520 (SIEX) und TMIK 1520 (SIEX). Während das MTES zum Test von MABS-Programmen vorgesehen ist, können mit TMIK CPU-Programme getestet werden.
Beide Testprogramme lassen sich nach ihrem Aufruf auf einen beliebigen Bereich des Hauptspeichers verschieben, außer in Betriebssystembereiche und in den Bildwiederholspeicher. Danach kann das zu testende Programm geladen werden.
MTES belegt 5 KByte und kann folgende Funktionen realisieren:
– Abarbeitung des zu testenden Programms bis zu einer vom Bediener vorgegebenen Stopstelle
– Anzeige oder Druck von Speicherbereichen
– Anzeige und Ändern von Speicherzellen
– Ausgabe getesteter Programme in die Phasenbibliothek von Diskette oder Kassette.
Als Stopstelle beim Programmtest können folgende Bedingungen festgelegt werden:
– Stop nach jedem MABS-Befehl (Schrittbetrieb)
– Stop nach Erreichen eines bestimmten Befehlszählerstandes
– Stop nach jedem Befehl mit einem vorgegebenen Operationscode
– Stop nach jedem Befehl, der einen vorgegebenen Speicherbereich verändert hat
– Stop nach jedem Sprungbefehl.
Eingelagerte CPU-Befehlsfolgen im zu testenden Programm werden im Echtzeitbetrieb abgearbeitet. In diese Folgen definierte Stopstellen werden ignoriert.
Im Schrittbetrieb wird auf dem Bildschirm jeder abgearbeitete Befehl angezeigt: Befehlszählerstand (Adresse des Befehls), Befehlsmnemonik und -code, ein wahlweise voreingestellter Protokollierbereich.
TMIK belegt etwa 3 KByte, es lassen sich folgende Funktionen von diesem Programm realisieren:
– Abarbeitung von CPU-Programmen im Echtzeitbetrieb bis zu einer beliebigen Stopstelle, die durch eine Adresse vorzugeben ist
– Schrittweises Abarbeiten von CPU-Programmen; es werden angezeigt: Befehlsadresse, -mnemonik, -code und alle CPU-Register
– Beliebige Manipulation der CPU-Register und Anzeige der Flags
– Anzeige und Änderung von Speicherzellen
– Druck von Speicherbereichen in hexadezimaler Darstellung.
Für Anzeigen und Eingaben werden die letzten drei Bildschirmzeilen genutzt. Eingelagerte Makrobefehlsfolgen werden auch im Echtzeitbetrieb abgearbeitet, in ihnen darf aber kein Stoppunkt definiert sein.

3.7.7.
Übertragung von Phasen auf Kassettenmagnetband

Die gesamte Programmentwicklung erfolgt unter Nutzung von Diskettendateien. Mit dem Kassettenmagnetbandgerät läßt sich keine Programmentwicklung durchführen. Damit aber Phasen auch vom Kassettenmagnetbandgerät aufgerufen werden können, ist ein Kopieren von Phasen von Diskette auf Kassette erforderlich. Dazu dient das Programm PHAK 1520 (SIEX). Neben der Hauptaufgabe des Kopierens einzelner Phasen oder der ganzen Phasenbibliothek werden noch folgende weitere Funktionen von PHAK realisiert:
– Initialisieren und Vorbereiten der Kassettenmagnetbänder
– Anzeige und Druck des Verzeichnisses der Phasenbibliothek der zu kopierenden Diskette.
Die Phasenbibliothek auf der Kassette ist folgendermaßen organisiert:
– Dateiformat COMPACT
– jede Phase wird eine COMPACT-Datei, wobei der Phasenname (auf der Diskette) zum Dateiname (auf der Kassette) wird
– kein Bibliotheksverzeichnis.
Vom Monitor aus können von einer derart organisierten Kassette die als Phase abgelegten Programme wie von einer Diskette aufgerufen werden.

Das Betriebssystem UDOS

Dr. Volkmar Köhler
VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt

4. UDOS

4.1. Systemübersicht

UDOS ist ein ausschließlich disketten-orientiertes Betriebssystem und spezifiziert den Bürocomputer auf das Haupteinsatzgebiet der Programmentwicklung. Die Diskettenarbeit ist auf einen möglichst schnellen Zugriff und eine effektive Ausnutzung des Diskettenplatzes orientiert. UDOS ist weniger gut geeignet für kommerzielle Problemlösungen, wie zum Beispiel die Behandlung großer Datenmengen oder Direktzugriffsdateien.

Die Programmentwicklungssoftware von UDOS ist auf folgende in der DDR entwickelte und produzierte Mikroprozessoren zugeschnitten:
- U 880 (mnemonische Schreibweise der Befehle analog dem SCP-Assembler)
- U 881/882 (Cross-Software)
- U 8000 (Cross-Software).

Die Programmentwicklung ist sowohl in Assemblerniveau als auch auf höherem Sprachniveau (vor allem PLZSYS, PLZASM) möglich.

Das Betriebssystem UDOS wird auf Grund seiner Zweckbestimmung für eine eingeschränkte Standardhardware geliefert. Dazu gehören:
- ZRE K 2526 mit 64 KByte RAM (ROM generell nicht erlaubt)
- 2 bis 3 Laufwerke Minifloppy K 5600.10
- Tastatur K 7634, K 7636 oder K 7637
- Monitor K 7222 oder K 7221
- Drucker SD 1152 oder SD 1157 (mit PIO- oder IFSS-Anschluß).

Nachrüstbar sind 8″-Floppy-Disk mit FM-Aufzeichnungsverfahren und Lochbandeinheit K 6200. Für das Lochband ist ein separater Treiber erforderlich.

Nicht im System implementiert sind die Geräte Kassettenmagnetband, 1/2″-Magnetband und Datenfernübertragung.

Für die älteren Versionen von UDOS (bis einschließlich Version 3.0) war das System automatisch ausgelegt für folgende Hardwarekomponenten:
- maximal 4 Laufwerke Minifloppy K 5600.10
- Tastatur K 7634 oder K 7636 (einschließlich der älteren Varianten K 7604 bzw. K 7606)
- Monitor K 7222 oder K 7221
- SD 1152 oder 1157 (mit PIO- oder IFSS-Anschluß).

Für weitere Hardwarekomponenten mußte entweder eine zweite Betriebssystemvariante (System mit integriertem 8″-Floppy) oder Treibersoftware nachgeladen werden (Lochband, logischer Druckertreiber, zweiter Floppy-Treiber für Mischkonfiguration Minifloppy/Standardfloppy u. ä.). Für bestimmte Hardwarekomponenten gab es noch keine Treibersoftware (K 7637).

Mit der generellen Einführung der seriellen Tastatur K 7637 in den Bürocomputerbestand wurde ein Umbruch in der Systemphilosophie von UDOS erforderlich. Damit war die Konfigurationspalette mittlerweile so breit geworden, daß eine automatische Anpassung des Betriebssystems an die konkrete Hardware nicht mehr tragbar war. Deshalb wurde für UDOS ab Version 4.0 ein Generiersystem (SG) geschaffen (wie es auch bereits für SIOS und SCP existierte), wodurch allerdings der Zustand entstand, daß für jeden Bürocomputer eine spezifische Systemdiskette erzeugt werden muß.

4.2. Systeminitialisierung

Nach Einschalten der Anlage bzw. RESET läuft automatisch der unter Punkt 1.4. skizzierte Vorgang ab. Bei UDOS sind aber einige Besonderheiten zu beachten.

Bei älteren Versionen des Betriebssystems (bis einschließlich V 3.0) wird durch den Systemlader nur der physische Teil (die gesamte Spur 2) eingelesen und zur Initialisierung angesprungen. Der dann erreichte Systemgrundzustand ist gleichzeitig die Bereitschaft zur Kommandoeingabe für den Systemdebugger. Das System meldet sich mit seinem Systemzeichen '+' (Systemprompt der physischen Ebene von UDOS). Zum residenten Teil des Betriebssystems gehört aber noch der logische Teil in Form von zwei weiteren Dateien:
- OS die eigentliche Executive, der Speicherverwalter und der logische Konsoltreiber
- ZDOS der logische Diskettentreiber.

Da die physische Ebene nicht in der Lage ist, Dateien zu behandeln (zu laden), muß per Bedienkommando 'O' (Organize) zunächst noch ein weiterer UDOS-spezifischer Lader (der sogenannte UDOS-Bootstrap-Lader) aktiviert werden. Dieser wird auf der Systemdiskette gesucht, geladen und angesprungen. Er selbst ist nun unter Zuhilfenahme des physischen Teils von UDOS in der Lage, die beiden Dateien OS und ZDOS zu laden. Anschließend wird der logische Teil durch einen Sprung in den Systemeintrittspunkt (auf Adresse 13DEH) initialisiert. Nach der betriebssystemspezifischen Initialisierung erfolgt noch automatisch eine anwendereigene Initialisierung. Dazu wird eine Kommandodatei mit dem reservierten Namen OS.INIT abgearbeitet. Der Inhalt von OS.INIT (d. h. die auszuführenden Kommandos) ist vom Anwender frei wählbar (über den Editor EDIT). Standardmäßig enthält OS.INIT die Kommandos, welche die Meldeausschriften des logischen Teiles von UDOS erzeugen. Nach Abarbeitung von OS.INIT meldet sich das logische System mit seinem neuen Promptzeichen '%'.

Bei neueren Versionen von UDOS (beginnend ab Version 4.0) entfällt in der Ladephase der Halt auf der physischen Ebene. Das Bedienerkommando 'O' wird intern erzeugt, so daß nach einem RESET unmittelbar aufeinanderfolgend der physische und der logische Teil von UDOS, also der gesamte residente Teil, geladen wird.

Mit der Ausgabe des Promptzeichens '%' gibt das System seine Bereitschaft zur Entgegennahme von Kommandos an. Jetzt sind aber nur noch Kommandos auf logischer Ebene erlaubt.

4.3. Dateiarbeit

UDOS arbeitet prinzipiell diskettenorientiert. Im System gibt es deshalb nur eine Dateiverwaltung für auf Disketten abgespeicherte Dateien (ZDOS). Es läßt

sich grundsätzlich auch denken, daß eine Dateiverwaltung auf anderen Datenträgern realisiert wird. Derartige logische Treiberprogramme (z. B. für Kassettenmagnetgerät oder Festplatte) werden vom Hersteller aber nicht vertrieben. Damit beziehen sich die nachfolgenden Ausführungen immer auf Disketten bzw. Diskettendateien.

4.3.1. Diskettenaufbau

UDOS arbeitet prinzipiell mit softsektorierten Disketten in Anlehnung an international gebräuchliche Normen, wie zum Beispiel ISO TC 97 Sc 11 Nr. 209 bzw. KROS 5108 und KROS 5110. Unterschiede bzw. Abweichungen gibt es in der logischen Formatierung und im physischen Sektoraufbau. Für die physische Sektorlänge ist bei UDOS nur der Wert 128 Byte zugelassen. Zusätzlich sind für die Dateiverwaltung aber an jeden Sektor sechs Byte angehängt (zu Lasten der nachfolgenden Lücke). Diese enthalten:
– 2 Byte Fore-Pointer (Zeiger auf den nächsten Satz)
– 2 Byte Back-Pointer (Zeiger auf den vorhergehenden Satz)
– 2 Byte CRC-Zeichen (Prüfzeichen für die Zeigerinformationen)

In allen anderen Punkten stimmt das UDOS-Format physisch mit dem SIOS-Datenaustauschformat überein.

Die Formatierung von UDOS-Disketten erfolgt mit dem Dienstprogramm FORMAT. Es realisiert folgende physische Formate:

- Standarddiskette (FM-Aufzeichnung):
77 Spuren mit je 26 Sektoren zu je 128 Byte (in der Summe 2002 Sektoren)
- Minidiskette (MFM-Aufzeichnung, einfache Spurdichte):
40 Spuren mit je 26 Sektoren zu je 128 Byte (in der Summe 1040 Sektoren)
- Minidiskette (MFM-Aufzeichnung, doppelte Spurdichte):
77 Spuren mit je 26 Sektoren zu je 128 Byte (in der Summe 2002 Sektoren).

Die Numerierung der Spuren beginnt bei 0 und endet bei n (n = Spuranzahl). Die Sektoren innerhalb einer Spur werden ebenfalls beginnend von 0 an bis 25 durchnumeriert (im Gegensatz zu SIOS und SCP: 1 ... 26!).

Werden bei der physischen Formatierung defekte Spuren erkannt, so werden diese einfach als belegt gekennzeichnet und so von weiteren Datenaufzeichnungen ausgeschlossen. Sie gehen damit zu Lasten der verfügbaren Nettokapazität der Diskette. Es können beliebig viele Spuren defekt sein (außer Spur 16H und 17H), trotzdem ist die Diskette unter UDOS nutzbar. Hierbei ist aber anzumerken, daß, wenn andere Ursachen ausgeschlossen sind, eine Diskette mit mehreren defekten Spuren offensichtlich verschlissen ist und aus Sicherheitsgründen nicht mehr benutzt werden sollte.

Mit FORMAT wird aber gleichzeitig noch eine logische Formatierung der Diskette vorgenommen, wobei einige für UDOS unerläßliche Informationen aufgezeichnet werden. Art und Umfang hängen davon ab, ob es sich um eine Arbeits- oder Systemdiskette handelt.

Arbeitsdiskette: Es sind zwei Komponenten erforderlich:
– DIRECTORY (Dateiverzeichnis)
Es belegt 11 festdefinierte Sektoren auf der Spur 16H.
– Diskettenbelegungsmap
Diese Tabelle belegt 3 festdefinierte Sektoren auf der Spur 17H und gibt für jeden Sektor der Diskette an, ob er belegt oder frei ist.

Systemdiskette: Eine Systemdiskette muß grundsätzlich erst einmal die gleichen Informationen enthalten wie eine Arbeitsdiskette, also DIRECTORY und Diskettenbelegungsmap. Zusätzlich sind folgende Informationen aufgezeichnet:
– Spur 0, Sektoren 0 ... 2 Systemlader neu (ab V. 3.0)
– Spur 1, Sektoren 0 ... 5 Systemlader alt (bei allen Versionen)
– Spur 2, Sektoren 0 ... 25 physischer Teil von UDOS
– Spur 15H, Sektoren 16 ... 21 UDOS-Bootstrap-Lader.

Damit sind bei einer Arbeitsdiskette a priori 14 Sektoren belegt und bei einer Systemdiskette 55.

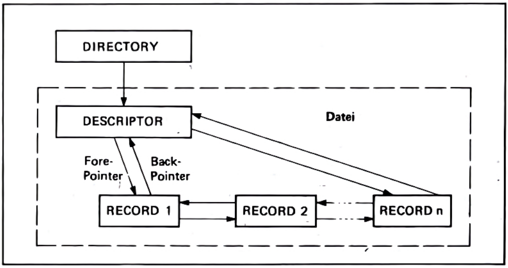
Eine komplette UDOS-Systemdiskette muß dann noch weitere Dateien (OS, ZDOS, OS.INIT und die im OS.INIT enthaltenen Kommandos) enthalten. Diese Dateien werden aber nicht mehr durch das FORMAT-Programm aufgezeichnet, sondern sind bei einem nachfolgenden Kopiervorgang zu überspielen.

4.3.2. Diskettendateien

Im Betriebssytem UDOS ist die Datei die Grundstruktur auf der Diskette. Das gilt sowohl für Daten als auch für Programme.

Eine UDOS-Datei besteht immer aus einem Kennsatz (DESCRIPTOR) und einer Anzahl von Datensätzen (RE-

Abb. 1 Dateiaufbau



logische Satzlänge (in Byte)		Anzahl der physischen
dezimal	hexadezimal	Diskettensektoren
128	80	1
256	100	2
512	200	4
1024	400	8
2048	800	16

Tab. 1 Zusammenhang zwischen logischer Satzlänge und physischer Sektorlänge

CORDs). Jeder Zugriff zu einer Datei erfolgt über den Dateinamen. Die Namen aller auf einer Diskette gespeicherten Dateien sind in einem Dateiverzeichnis, dem DIRECTORY, enthalten. Die Zusammenhänge soll Abb. 1 verdeutlichen. Das DIRECTORY ist selbst eine Datei. Es enthält je Datei folgende Eintragung:
– 1 Byte Länge des Dateinamens
– n Byte Dateiname (entsprechend der Länge)
– 2 Byte Diskettenadresse des Dateikennsatzes.

Dateiaufzeichnung

Im Betriebssystem UDOS gibt es bezüglich der Abspeicherung von Dateien einige Besonderheiten zu beachten. Die Datensätze einer Datei, die eine einheitliche logische Satzlänge besitzen, bilden physisch kein zusammenhängendes Datenmassiv, sondern sind vielmehr über die Diskette „verstreut". Die physische Abspeicherung wird vom System automatisch organisiert (dafür ist ZDOS verantwortlich), wobei nach Kriterien für einen möglichst schnellen Zugriff und das restlose Ausnutzen des verfügbaren Platzes auf der Diskette optimiert wird. Demgegenüber bildet ein Datensatz (RECORD) immer eine physische Einheit, das heißt eine zusammenhängende Anzahl von Diskettensektoren Tab. 1. So gilt zum Beispiel bei einer Satzlänge RL = 400H, daß jeder RECORD dieser Datei aus acht aufeinanderfolgenden Sektoren besteht (innerhalb einer Spur, da ein RECORD nicht über eine Spurgrenze hinausgehen darf). Da aus bestimmten Gründen als Satzlänge nur Zweierpotenzen größer als 128 in Frage kommen, ergibt sich logischerweise als maximale Satzlänge 800H. Die nächstmögliche Zweierpotenz ist 1000H bzw. 32 Sektoren. Dies würde aber auf jeden Fall über die Spurgrenze hinausgehen, da in einer Spur einer softsektorierten Diskette nur 26 Sektoren angeordnet sind.

Infolge der „gestreuten" und scheinbar ungeordneten Aufzeichnung der RECORDs einer Datei auf der Diskette macht sich ein Mechanismus erforderlich, der die eindeutige Zuordnung von RECORDs zu einer Datei gewährleistet. Dieser Mechanismus wird mittels ZDOS über die im Abschnitt 4.3.1. erwähnten Zeiger und die Diskettenbelegungsmap realisiert. Durch die Zeiger sind alle zu einer Datei gehörenden RECORDs logisch miteinander verkettet, und zwar in zweifacher Hinsicht (vorwärts und rückwärts). Die Zeiger sind sogar an jeden Sektor angehängt. Es gilt, daß sie innerhalb eines RECORDs gleich sind. Ausgewertet werden aber immer nur die des jeweilig letzten Sektors eines RECORDs.

Dateibezeichnung und Dateiaufruf

Zur eindeutigen Identifikation einer Datei dient ihr Name. Es gelten dafür folgende Konventionen:
– Der Dateiname hat eine Länge zwischen 1 und 32 ASCII-Zeichen.
– Das erste Zeichen muß ein Buchstabe sein.
– Punkt '.' und Unterstreichung '_' dienen der Namenserweiterung, Leerzeichen sind nicht zugelassen.

Bestimmte Systemprogramme erzeugen bzw. fordern festgelegte Namenserweiterungen, die hinter dem Punkt im Dateinamen stehen. Einige reservierte Namenserweiterungen bedeuten:

– Dateiname.S	Quellprogramm
– Dateiname.OBJ	Objektprogramm
– Dateiname.OLD	Back-up-Kopie
– Dateiname.L	Druckdatei einer Übersetzungsliste
– Dateiname.SYM	Symboldatei.

Ansonsten besitzt der Programmierer alle Freiheiten über die Namenserweiterung.

Innerhalb eines Dateinamens besitzt der Stern '*' eine Sonderbedeutung. Er ersetzt ab dieser Position im Dateinamen eine beliebige Zeichenkette. Beispiele dafür sind (siehe auch SCP, Punkt 5.3.5.):

DAT*	alle Dateien, deren Namen mit DAT beginnen
*.L	alle Listingdateien
*.OLD	alle Back-up-Kopien
*	alle Dateien (= beliebiger Dateiname).

Beim Aufruf einer Datei sind mehrere Informationen erforderlich, da sich die Datei auf unterschiedlichen Datenträgern und Laufwerken befinden kann. Es sind deshalb – sofern keine Standardannahmen gelten sollen – das Laufwerk und das notwendige Dateiorganistions- und -verwaltungsprogramm (der logische Treiber) zusätzlich zum Dateinamen zu spezifizieren. Da es bei allen bisher vertriebenen UDOS-Versionen nur einen Datenträger mit Dateiverwaltung (Diskette) gibt, kann bei Aufrufen von Diskettendateien dieser Parameter (ZDOS) entfallen, da solche nicht vollständig spezifizierten Dateirufe automatisch dorthin geleitet werden. Für den Dateiaufruf gilt deshalb folgendes Format:

¤Treiber:Laufwerksnummer/Dateiname

Das Währungszeichen '¤' muß vor dem Namen des Dateiverwaltungsprogramms stehen, dahinter steht der Doppelpunkt ':', falls noch weitere Angaben folgen. Das Laufwerk ist vom Dateinamen durch einen Schrägstrich '/' zu trennen.

Beispiele:

¤ZDOS:2/FORMAT
Es wird die Datei FORMAT auf Diskette auf Laufwerk 2 gesucht und aufgerufen.

FORMAT
Es wird die Datei FORMAT auf Diskette (vor-

eingestellt!) auf allen aktiven Laufwerken gesucht und – wenn das erstemal gefunden – aufgerufen.

□SD
Das Gerät SD (Drucker) wird angesprochen; wenn z. B. eine Datei gedruckt werden soll, ist der Dateiname für den Drucker nicht relevant.

Dateityp und Dateiattribute
Es gibt bei UDOS vier mögliche Dateitypen, wovon aber einer (Typ DIRECTORY) von untergeordneter Bedeutung ist. Der Typ wird kodiert im Kennsatz eingetragen und von verschiedenen Dienst- und Hilfsprogrammen geprüft. Er wird üblicherweise mit einem Buchstaben bezeichnet:

A ASCII
Textdatei im ASCII-Format
B BINARY
Datei mit Binärinformationen
(z. B. Objektdatei)
P PROCEDURE
abarbeitbares Programm (Maschinencode)
D DIRECTORY
Diesen Typ darf nur die Datei DIRECTORY besitzen (ist genau einmal auf jeder Diskette vorhanden).

Zusätzlich zum Typ läßt sich noch ein Subtyp definieren. Dazu dient eine Zahl im Bereich zwischen 1 und 15. Erforderlich ist dieser Subtyp zum Beispiel für alle Treiberprogramme. Das OS verlangt für jeden in das System einzubindenden Treiber den Typ P1.
Jeder Datei lassen sich Attribute zuordnen, die die Anwendungsbedingungen beschreiben. Es gibt sechs solcher Attribute, wovon gleichzeitig auch mehrere spezifiziert sein können:

W write protected
Schreibgeschützt; die Datei kann weder überschrieben noch gelöscht werden.
E erase protected
Löschgeschützt; die Datei kann nicht gelöscht, ihre Sätze aber überschrieben werden.
L locked
Blockiert gegen Attributänderungen.
S secret
Geheim; die Datei wird bei einem normalen Aufruf nicht gefunden.
R random
Wahlfreier Zugriff erlaubt.
F force

Byte	Inhalt
0 ...5	reserviert
6 ...7	Zeiger zum DIRECTORY-Sektor mit dem Dateinamen
8 ...9	Zeiger zum ersten RECORD
10 ...11	Zeiger zum letzten RECORD
12	Dateityp (verschlüsselt in Bit 7 ...4 in der Reihenfolge P, D, A, B)
13 ...14	Anzahl der RECORDs in der Datei
15 ...16	logische Satzlänge
17 ...18	reserviert
19	Dateiattribute (verschlüsselt in Bit 7 ...2 in der Reihenfolge W, E, L, S, R, F)
20 ...21	Startadresse bei PROCEDURE-Dateien
22 ...23	Anzahl der Bytes im letzten RECORD
24 ...31	Dateierstellungsdatum
32 ...39	Datum der letzten Änderung
40 ...(40+4xn)	n = Segmentanzahl mit $0 \le u \le 16$ je Segment steht hier die Segmentanfangsadresse und die Segmentlänge; nach dem letzten Segment stehen 4 Byte OH.
122 ...123	kleinste Segmentanfangsadresse (LOW_ADDRESS)
124 ...125	größte Segementenadresse (HIGH_ADDRESS)
126 ...127	Stackgröße

Anmerkung: Die Angaben ab Byte 40 sind nur für PROCEDURE-Dateien relevant!

Tab. 2 Dateikennsatz UDOS (DESCRIPTOR)

Beim Laden der Datei wird keine Rücksicht auf die aktuelle Speicherbelegung genommen.

Attribute können mit dem externen Kommando SET (siehe 4.5.3.) geändert werden, sofern nicht das Attribut L spezifiziert ist.

Dateikennsatz (DESCRIPTOR)
Wenn eine Datei aufgerufen werden soll, so wird ihr Name im Dateiverzeichnis (DIRECTORY) auf der Diskette gesucht. Als wesentliche Information ist dem Dateinamen die Diskettenadresse des Kennsatzes zugeordnet, das heißt, die Dateikennsätze befinden sich nicht auf reservierten Plätzen auf der Diskette, wie zum Beispiel bei SIOS, sondern, sie können sich an beliebigen Stellen befinden. Sie sind immer in Nähe des ersten RECORDs untergebracht.
Der Dateikennsatz enthält nun alle Informationen, die die Struktur *Datei* umfassend beschreiben (Ausnahme: Dateiname). Wichtig sind vor allen Dingen die Zeiger zum ersten bzw. letzten Satz (d. h. Anfang und Ende der Datei), die logische Satzlänge und Satzanzahl. Mit diesen Angaben und der logischen Verkettung der RECORDs durch die Zeigerorganisation lassen sich alle Daten der Datei eindeutig lokalisieren.
Der genaue Aufbau eines DESCRIPTORs ist in Tab. 2 beschrieben.

4.4. Arbeitsspeicheraufteilung

UDOS benötigt zwar nicht zwingend 64 KByte Arbeitsspeicher, aber bei einer kleineren Ausbaustufe ist mit Einschränkungen bezüglich der Lauffähigkeit von Komponenten der höheren Software zu rechnen. Deshalb wird bei den folgenden Betrachtungen immer von 64 KByte ausgegangen. Die im SIOS verfügbare Speichererweiterung von 48 KByte ist unter UDOS nicht nutzbar.
Für die Speicherbelegung gelten bei UDOS folgende Prämissen:

- Es ist kein ROM im Arbeitsspeicher zulässig.
- Der Bildwiederholspeicher ist wie bei SIOS und SCP adreßmäßig am Ende des Arbeitsspeichers eingeordnet. Bei älteren Betriebssystemversionen (bis Version 3.0) ist er 1 oder 2 KByte groß, und seine Anfangsadresse liegt bei 0FC00H bzw. 0F800H. Ab Version 4.0 sind Größe und Lage des Bildwiederholspeichers generierbar.
- Unmittelbar vor dem Bildwiederholspeicher benötigt das System einen variablen Bereich für die Diskettenbelegungsmaps. Für jedes aktive Laufwerk (d. h. tatsächlich eingelegte Diskette) werden 3 × 128 Byte belegt. Da die Anzahl der benötigten Disketten während der Abarbeitung eines Programms wechseln kann, hat auch dieser Arbeitsspeicherbereich keine konstante Größe.
- Der residente Teil von UDOS befindet sich in den ersten 16 KByte des Arbeitsspeichers.

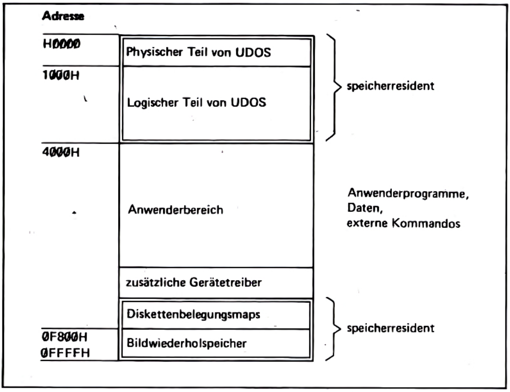


Abb. 2 Arbeitsspeicheraufteilung bei UDOS

Abb. 2 zeigt schematisch die Arbeitsspeicheraufteilung. Eine weitere Besonderheit gegenüber SIOS und SCP ergibt sich dadurch, daß mit dem residenten Teil nur eine ganz bestimmte Peripheriekonfiguration bedient wird. Da aber UDOS auf das Einbinden weiterer Peripherietreiber geradezu zugeschnitten ist, muß das bei Arbeitsspeicherbetrachtungen berücksichtigt werden. Diese zusätzlichen Treiber sind zweckmäßigerweise vor den Diskettenmaps einzuordnen, um den verbleibenden Arbeitsspeicherbereich möglichst zusammenhängend nutzen zu können. Das bedeutet, daß bei einer Aufstockung der Konfiguration der für den Anwender verfügbare Speicherbereich kleiner wird. Der Bereich von 4000H bis zur ersten Diskettenbelegungsmap steht dem Anwender für eigene Programme sowie für alle nachladbaren Dienst- und Hilfsprogramme und Programmaufbereitungskomponenten (Programmiersprachen) zur Verfügung. Dieser Bereich ist bei der im Abschnitt 4.1. angegebenen Standard-Konfiguration etwa 45 KByte groß.

4.5. Kommandosystem

UDOS verfügt über interne und externe Kommandos. Die internen Kommandos sind integraler Bestandteil der Executive (Datei OS) und damit nach dem Laden des vollständigen Systems speicherresident und sofort abarbeitbar. Diese Gruppe ist relativ klein und erfüllt systematische Aufgaben. Die externen Kommandos stehen als Programmdateien (Dateityp PROCEDURE) auf dem Datenträger und werden bei Bedarf in den Anwenderbereich des Arbeitsspeichers geladen und abgearbeitet. Nach Beendigung der Routine wird der von ihnen belegte Speicherplatz wieder freigegeben.

4.5.1. Kommandointerpreter

Der Kommandointerpreter ist Teil der Executive und immer dann aktiv, wenn auf der Konsole das Systemprompt '%' erscheint. In diesem Falle erwartet er die Eingabe einer Kommandokette durch den Bediener.

Eine Kommandokette darf maximal 256 Zeichen lang sein (einschließlich Endezeichen 'CR' = ODH). Normalerweise gelangt sie von der Tastatur in den Eingabepuffer INBU des Systems, sie kann aber auch programmtechnisch dort eingeschrieben werden. Als Trennzeichen zwischen den Kommandos dient das Semikolon ';'. Mit dem Zeichen 'CR' wird die Eingabe abgeschlossen und die Kette von vorn beginnend abgearbeitet. Die prinzipielle Arbeitsweise ist in Form eines Programmablaufplanes in Abb. 3 dargestellt.

Wichtig ist noch zu wissen, daß ein Komma, welches unmittelbar hinter dem Namen eines externen Kommandos steht, bewirkt, daß dieses Programm nur geladen, aber nicht gestartet wird. Außerdem wird nach Ausführung eines externen Kommandos der von diesem Programm belegte Speicherplatz zwar wieder freigemeldet, das Programm selbst aber nicht gelöscht. Es kann deshalb sofort erneut ausgeführt werden, ohne daß es noch einmal geladen werden muß (siehe auch 4.5.2., Kommando 'X').

4.5.2. Interne Kommandos

Zur eindeutigen Identifikation reicht es, jeweils nur die signifikanten Zeichen des Kommandonamens (nachfolgend durch Großbuchstaben dargestellt) einzugeben. Es folgt eine Kurzbeschreibung der internen Kommandos:

- *Initialize*

Es bewirkt insbesondere, daß von allen eingelegten Disketten die Belegungsmaps in den Arbeitsspeicher gelesen werden. Damit ist dieses Kommando bei jedem Diskettenwechsel zwingend notwendig, vor allem, wenn auf die Diskette geschrieben werden soll. Ansonsten kann es zu Datenverlusten kommen.

- *Close*

Das als Parameter angegebene logische Gerät (Datei) wird geschlossen.

- *Allocate und DEAllocate*

Mit Allocate kann ein durch Parameter spezifizierter Speicherbereich reserviert,

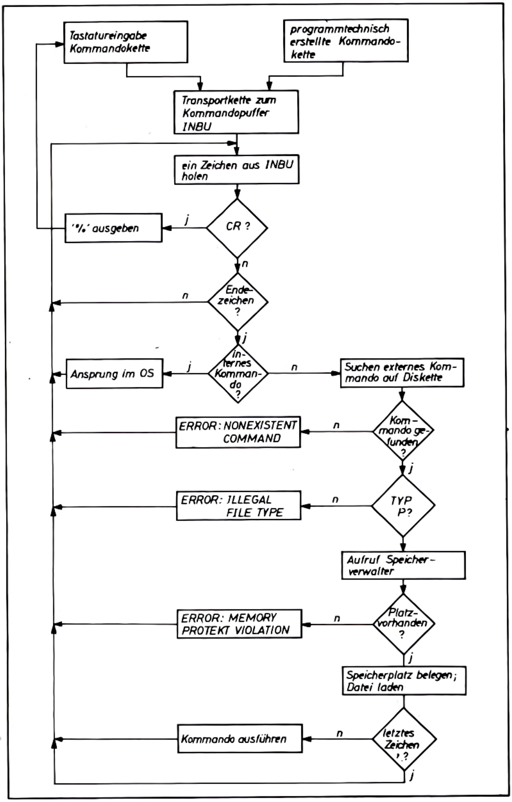


Abb. 3 Schematische Darstellung der Arbeit des Kommandointerpreters

mit DEAllocate eine solche Reservierung wieder gelöscht werden.

- *Release*

Es wird der gesamte Anwenderspeicher wieder freigegeben.

- *eXecute*

Das zuletzt geladene Kommando wird erneut ausgeführt bzw. es wird an die als Parameter angegebene Adresse gesprungen. Soll mit 'X' ein externes Kommando ausgeführt werden, das Parameter erfordert, so ist zunächst nach dem X ein Stern '*' einzugeben und anschließend die Parameter.

Beispiel:

Es wird MOVE geladen, anschließend werden Disketten gewechselt und danach sollen alle ASCII-Dateien vom Laufwerk 1 nach 2 transportiert werden.

Es sind folgende Kommandos einzugeben:

%MOVE,
%I
%X * P=& T=A S=1 D=2
%

- *Brief und Verbose*

Damit werden zwei Betriebsarten des logischen Konsoltreibers eingestellt. Im Verbose-Modus wird die eingegebene Zeichenkette ein zweites Mal als Echo auf der Konsole ausgegeben, im Brief-Modus entfällt das Echo.

- *Force*

Wenn ein Kommando (bzw. beliebiges Programm vom Typ P) in den Arbeitsspeicher geladen werden soll, so muß der dafür erforderliche Speicherplatz im Speicherverwalter frei gemeldet sein, anderenfalls erfolgt die Fehlermeldung "MEMORY PROTECT VIOLATION". Mit dem Kommando Force kann man sich über alle Reservierungen hinwegsetzen und das Laden eines Programmes erzwingen.

- *':'*

Der hinter dem Doppelpunkt anzugebende hexadezimale arithmetische Ausdruck wird berechnet und auf der Konsole angezeigt. Es sind nur die Operationen (+ – × /) erlaubt. Klammern sind nicht zugelassen.

- *Debug*

Damit sind der Systemdebugger des physischen Teils von UDOS aufgerufen. Er meldet sich mit dem Promptzeichen '+' und gestattet die Ausführung von elementaren Kommandos für die Programmtestung, Speicher- und Registeranzeige bzw. -modifikation, Portein- und Portausgabe sowie physische Diskettenarbeit. Detaillierte Informationen zum Systemdebugger sind in Abschnitt 4.6.2. enthalten.

4.5.3. Externe Kommandos

Externe Kommandos stehen als PROCEDURE-Dateien auf dem Datenträger und müssen deshalb beim Aufruf durch ihren vollständigen Dateinamen lokalisiert werden. Eine Abkürzung wie bei den internen Kommandos ist nicht möglich. Die nachfolgende Auswahl

enthält Kurzbeschreibungen der Standardkommandos von UDOS.

FORMAT

Dieses Kommando wird zum Einrichten neuer UDOS-Disketten benutzt. Es kann sowohl Arbeits- als auch Systemdisketten formatieren und bringt die dafür erforderlichen Informationen auf die Diskette. Es ist in der Lage, alle bei UDOS zugelassenen Diskettenformate zu erzeugen (also für 8″ FM-Aufzeichnung, für 5,25″ mit 40 Spuren und für 5,25″ mit 77 Spuren). Dieses Kommando besitzt keine Parameter. Die zur Formatierung neuer Disketten erforderlichen Angaben werden abgefragt. Bei der Formatierung festgestellte fehlerhafte Spuren werden als belegt markiert (in der Diskettenbelegungsmap) und angezeigt.

STATUS

STATUS ermittelt die Anzahl der freien und der belegten Sektoren je Diskette (= Diskettenstatistik) und gibt diese für alle eingelegten Disketten auf der Konsole aus. Falls als Parameter eine Laufwerksnummer (0 ... 3) angegeben ist, so wird die Diskettenstatistik nur für dieses Laufwerk ermittelt. Für jede Diskette wird noch der Diskettenname (ID) mit angezeigt.

MOVE, COPY, COPY.DISK

Diese drei Kommandos dienen dem Kopieren. Mit MOVE können Dateien bzw. ganze Dateigruppen, mit COPY eine einzelne Datei und mit COPY-.DISK eine gesamte Diskette kopiert werden. Im einzelnen gilt:

- *%MOVE Dateibezeichnung Options*

Die Dateibezeichnung kann vollständig (entspricht einer Datei) oder teilweise spezifiziert (Dateigruppe) sein. Die Options umfassen zum Beispiel Dateityp, Dateiattribute, Erstellungsdatum oder Datum der letzten Änderung (siehe auch CAT).

Quellen und Ziellaufwerk werden durch 'S=Laufwerksnummer' und 'D-Laufwerksnummer' spezifiziert (die Richtung von Laufwerk 0 nach Laufwerk 1 ist immer voreingestellt!)

Beispiel:

%MOVE *.S S=1 D=3 P=&

Es werden alle Dateien, deren Namen mit '.S' enden, von der Diskette im Laufwerk 1 auf die Diskette im Laufwerk 3 kopiert.

- *%COPY Datei1 Datei2 Options*

Datei1 und Datei2 können Dateinamen oder vollständige Dateispezifikationen (incl. Laufwerksnummer und Treiber) sein. Mit diesem Kommando wird Datei1 gelesen und als Datei2 abgelegt, wobei die Eigenschaften von Datei1 übernommen werden, sofern die Options nichts anderes bestimmen. Mit den Options lassen sich zum Beispiel die Recordlänge oder der Dateityp neu festlegen.

Beispiel:

%COPY 1/FORMAT 0/INIT RL=200

Das Programm FORMAT von Laufwerk 1 wird auf das Laufwerk 0 kopiert und dort unter dem Namen INIT mit der neuen Recordlänge von 200H abgelegt.

%COPY 2/TEXT ◻SD

Die Datei mit dem Namen Text von Laufwerk 2 wird zum Drucker kopiert (d.h. gedruckt). Dabei muß der Treiber SD vorher aktiviert worden sein (siehe ACTIVATE).

- *%COPY.DISK 'Quellaufwerk' TO 'Ziellaufwerk' Option*

Kopiert die Diskette des Quellaufwerkes auf die Diskette des Ziellaufwerkes. Wenn die Laufwerksangaben fehlen, ist die Richtung 0→1 voreingestellt. Als Option ist 'V' zugelassen, wodurch ein Kontrollesen der beschriebenen Diskette ausgeführt wird.

Beispiel:

%COPY.DISK 3 TO 1 V

Kopiert die Diskette von Laufwerk 3 auf die Diskette im Laufwerk 1 mit zusätzlichem Kontrollesen.

CAT, EXTRACT

Diese Kommandos dienen der Ausgabe von Informationen des DIRECTORYs und der einzelnen DESCRIPTORen. Allgemein gilt:

- *%CAT Options*

Zu den wichtigsten Options gehören der Dateiname, das Darstellungsformat und die Spezifikation bestimmter DESCIPTOR-Informationen.

Für den Dateinamen gilt das unter Punkt 4.3.2. Gesagte. Fehlt er völlig, dann ist das gleichbedeutend mit '*'. Fehlt die Formatangabe, dann bedeutet das Kurzformat. Dabei werden nur Dateiname und Laufwerksnummer ausgegeben. Ist als Format F=L spezifiziert, dann erfolgt die Ausgabe mit Kennsatzinformationen. Dazu gehören neben Dateiname und Laufwerksnummer der Dateityp, Recordanzahl und -länge, die Dateiattribute, der Eintrittspunkt bei PROCEDURE-Dateien sowie das Datum der Erstellung und der letzten Änderung.

Beispiel:

%CAT

Alle nicht geheimen Dateien aller Disketten werden im Kurzformat auf der Konsole (genauer: logisches Gerät 3) ausgegeben.

%CAT * P=WS F=L T=A D=2

Alle ASCII-Dateien (T=A) der Diskette in Laufwerk 2 (D=2), die die Eigenschaften W und S besitzen (P=WS), werden im Langformat aufgelistet.

%CAT *.S *.L CDATE<=830101

Alle Quell- und Listingdateien, die vor dem 1.1.83 erstellt wurden, werden im Kurzformat aufgelistet, sofern sie nicht geheim sind.

- *%EXTRACT Dateiname*

Es werden die Kennsatzinformationen der angegebenen Datei ähnlich dem Kommando CAT (Langform) ausgegeben. Dazu gehören wieder Recordanzahl und -länge, die Anzahl Bytes im letzten Record sowie bei PROCEDURE-Dateien Eintrittspunkt, kleinste und größte Speicheradresse, Stackgröße und Segmentgrößen.

PRINT, DUMP

Beide Kommandos dienen der Anzeige von Dateien. Mit PRINT können nur ASCII-Dateien, mit DUMP beliebige Dateien, jedoch nur in Hexform angezeigt werden. Bei beiden ist als Parameter ein Dateiname zu spezifizieren.

RENAME, DELETE, SET

Mit RENAME kann eine Datei umbenannt werden:

%RENAME 'alter Dateiname' 'neuer Dateiname'

DELETE dient dem Löschen von Dateien auf Diskette (falls diese nicht die Attribute W oder E besitzen). Damit kann freier Platz auf der Diskette geschaffen werden:

%DELETE Dateibezeichnung Options

```
                    MEMORY ALLOCATION MAP

           0             400           800           C00
           |   |         |   |         |   |         |   |
0000       AAAAAAAA      AAAAAAAA      AAAAAAAA      AAAAAAAA
1000       AAAAAAAA      AAAAAAAA      AAAAAAAA      AAAAAAAA
2000       AAAAAAAA      AAA.AAAA      AAAAAAAA      AAAAAAAA
3000       AAAAAAAA      AAAAAAAA      AAAAAAAA      AAAAAAAA
4000       AAAAA...      ........      ........      ........
5000       ........      ........      ........      ........
6000       ........      ........      ........      ........
7000       ........      ........      ........      ........
8000       ........      ........      ........      ........
9000       ........      ........      ........      ........
A000       ........      ........      ........      ........
B000       ........      ........      ........      ........
C000       ........      ........      ........      ........
D000       ........      ........      ........      ........
E000       ........      ........      ........      ........
F000       ........      .....AAA      AAAAAAAA      AAAAAAAA

        "." => FREE        "A" => ALLOCATED
%
```

Legende:

- (gestrichelter Rahmen) Programm DISPLAY
- (Rahmen) Diskettenbelegungsmap für eine Diskette
- (strichpunktierter Rahmen) Bildwiederholspeicher für K 7222

Abb. 4 Arbeitsspeicherbelegung (Kommando DISPLAY)

Die Dateibezeichnung kann vollständig oder teilweise spezifiziert sein. Bevor eine Datei gelöscht wird, ist das vom Bediener ausdrücklich zu quittieren. Die Options sind mit denen von CAT weitgehend identisch.

Das Kommando SET dient der Einstellung verschiedener Systemparameter bzw. Kennsatzinformationen:

%SET Options

Beispiel:

%SET PROPERTIES OF Dateiname TO WLS

Die Datei bekommt die Attribute WLS, falls nicht schon vorher L spezifiziert war.

%SET DISKCON=5558

Die Floppykonfiguration wird eingestellt; die Laufwerke 0, 1 und 2 als 5,25"-Laufwerk mit 40 Spuren, das Laufwerk 3 als 8"-Laufwerk mit FM-Aufzeichnung.

IMAGE

Mit IMAGE läßt sich ein durch Options definierter Speicherbereich als UDOS-Datei (Typ P) auf Diskette ablegen. Der Speicherbereich muß nicht zusammenhängend sein, sondern kann aus maximal 16 Segmenten bestehen:

%IMAGE Dateiname Options

Beispiel:

%IMAGE TEST 4000 4FFF E=4100 RL=200

Auf Diskette wird eine Datei mit dem Namen TEST in der Recordlänge 200H aufgezeichnet, deren Inhalt mit dem Speicherbereich von 4000H bis 4FFFH identisch ist. Der Eintrittspunkt liegt bei Adresse 4100H.

COMPARE

Der Inhalt zweier Dateien wird byteweise miteinander verglichen:

%COMPARE Datei1 Datei2

Jede Abweichung wird ausgegeben. Wenn keine Meldung erfolgt, liegt Identität vor.

DO

Mit DO können mehrere Kommandos, die als Datei auf Diskette stehen, automatisch abgearbeitet werden. Es können auch Parameter spezifiziert werden. Die Funktionsweise ist analog dem Kommando SUBM unter SCP (siehe 5.5.3.). Die Kommandodatei kann mit dem Editor erzeugt werden.

DISPLAY

Dieses Kommando bezieht sich direkt auf den Speicherverwalter der Executive (OS). Es besitzt keine Parameter und bringt die aktuelle Speicherreservierung des Arbeitsspeichers zur Anzeige (siehe Abb. 4).

DATE

Im System (OS) wird ein Systemdatum verwaltet. Mit DATE kann es abgefragt (ohne Parameter) oder neu gesetzt werden (mit Parameter).

Beispiel:

%DATE 851001

Als Systemdatum wird der 1. 10. 85 eingestellt, es erfolgt die Meldung:

TUESDAY, OCTOBER 1., 1985

ACTIVATE, DEACTIVATE, DEFINE, MASTER, LADT

Diese Kommandogruppe bezieht sich auf das logische E/A-System von UDOS (siehe auch 4.6.1.).

Mit ACTIVATE werden Gerätetreiber aktiviert (in das E/A-System eingebunden), mit DEACTIVATE wieder herausgelöst. LADT listet alle aktiven Geräte auf Konsole auf. Mit DEFINE wird einem aktiven Gerät eine logische Nummer zugeordnet (im Prinzip entspricht das der Zuordnung einer Datei zu einem Gerät) bzw. läßt sich die voreingestellte Zuordnung ändern.

Mit dem MASTER-Kommando kann abgefragt werden, welches Gerät im Moment Master ist bzw. ein neues Mastergerät definiert werden.

ECHO, PAUSE, RESTORE_TABS, SET_TABS

Diese Kommandogruppe wirkt auf den logische Konsoltreiber CON (Bestandteil der Datei OS).

ECHO hat als Parameter eine Zeichenkette und bewirkt deren Ausgabe auf die Konsole. Mit PAUSE wird die Abarbeitung weiterer Kommandos unterbrochen. Erst mit einer erneuten Tastenbetätigung werden die nach PAUSE in der Kommandozeile stehenden Kommandos abgearbeitet.

Die beiden anderen Kommandos haben als Parameter einen Dateinamen. Mit SET_TABS wird die aktuelle Tabulatorbelegung der Konsole unter diesem Da-

teinamen auf Diskette abgelegt, mit RESTORE_TABS wird die aktuelle Tabulatorbelegung der Konsole durch die der spezifizierten Datei ersetzt.

4.6. E/A-System

Das E/A-System von UDOS besteht aus einem physischen und einem logischen Teil.

Der physische Teil realisiert die gerätenahen Funktionen. Obwohl ausgewählte Unterprogramme dieser Ebene vom Anwender genutzt werden können, ist sie für den Anwender nicht die Hauptebene. Vielmehr steht diese Programmierebene dem Systemprogrammierer bzw. im Havariefall dem Anwender zur Verfügung.

Der logische Teil des E/A-Systems bietet eine hohe E/A-Flexibilität. Eine standardisierte E/A-Schnittstelle im OS ermöglicht eine einfache Implementierung zusätzlicher Treiber. Über komplexe E/A-Befehle (I/O-Requests) kann diese Schnittstelle einfach bedient werden. Diese Ebene bietet dem Anwender einen hohen Komfort, da eine Bedienung sowohl über die Assemblersprache als auch über die höhere Programmiersprache PLZ möglich ist. Damit wird die logische E/A-Ebene zur Hauptarbeitsebene für den Anwenderprogrammierer.

4.6.1. Logisches E/A-System

Grundsätzliches

Zum logischen E/A-System gehören standardmäßig zwei Treiber:
- ZDOS logischer Floppytreiber
- CON logischer Konsoltreiber (als Bestandteil des OS).

Beide Treiber werden über die bereits erwähnte Standard-E/A-Schnittstelle bedient. Diese Schnittstelle ist in Form eines Parametervektors aufgebaut, dessen Format Tab. 3 zeigt.

Bei UDOS gibt es auch wie bei SIOS oder SCP komplexe E/A-Befehle bzw. -Rufe (sog. I/O-Requests). Diese Requests beziehen sich immer auf einen solchen Vektor. Sie werden als ein CALL zum Systemeintrittspunkt (Adresse 1003H) umgesetzt mit dem eben genannten Vektor als Parameter. Zu diesem Zweck muß vorher das IY-Register der CPU mit der Anfangsadresse des I/O-Vektors geladen werden. Das OS untersucht bei einem solchen Request den ersten Parameter und leitet den gesamten Ruf dann zu dem der logischen Geräteadresse (Byte 0) zugeordneten Treiber.

Das E/A-System ist in der Lage, mit maximal 20 logischen Geräten zu arbeiten. Jedes logische Gerät muß, bevor es einen Request bearbeiten kann, aktiviert sein und auch tatsächlich eine logische Geräteadresse (UNIT) besitzen. Im E/A-System sind diese Informationen in einer Tabelle der logischen Geräte (active device table = ADT) gespeichert. Diese Tabelle kann man mit dem Kommando LADT anzeigen. Ihr Aussehen nach der Initialisierung zeigt Tab. 4. Die Konsole CON erhält die UNIT 1 (CONIN Tastatureingabe), UNIT 2 (CONOUT Bildschirmausgabe) und UNIT 3 (SYSLST Bildschirm- oder Druckerausgabe) zugeordnet. ZDOS als logischer Floppytreiber wird mit den Adressen 4 bis 19 versehen. UNIT 20 wird dem NULL-Treiber (vergleichbar mit einem NOP-Befehl, das heißt, diesbezügliche Requests bewegen keine Daten) zugeordnet.

Die ADT ist keine starre Tabelle, sondern kann den unterschiedlichen E/A-Bedingungen angepaßt werden. Mit dem Kommando ACTIVATE kann ein neuer Treiber hinzugefügt werden, mit DEACTIVATE kann einer gestrichen werden. Mit dem Kommando DEFINE lassen sich die Zuordnungen von Gerätetreibern und logischen Geräteadressen ändern. Letzteres läßt sich auch programmtechnisch durch einen sogenannte ASSIGN-Request erreichen.

Die Verbindung des E/A-Systems mit Anwenderdateien erfolgt durch den OPEN-Request, der für jede Datei im Anwenderprogramm programmiert sein muß. Mit CLOSE wird eine derartige Verbindung wieder gelöst. Schematisch ist dieser Vorgang in Abb. 5 gezeigt. Die bei System-CALL dem jeweiligen Treiber übergebenen Requests müssen dann in physische Aktionen des betreffenden Gerätes umgesetzt werden. Natürlich wird nicht jeder Treiber alle im System möglichen Requests umsetzen. So sind zum Beispiel Lese-Requests bei einem Druckertreiber unangebracht, da der Drucker lediglich Informationen ausgeben kann. Somit ist der Umfang der zu realisierenden Requests von der Spezifik des jeweiligen E/A-Gerätes abhän-

Byte	Inhalt
0	logische Geräteadresse (UNIT)
1	Requestkode (gibt die auszuführende Operation an)
2, 3	Anfangsadresse des Datenbereiches
4, 5	Länge des Datenbereiches
6, 7	Rückkehradresse für Interruptbetrieb (z. Z. nicht benutzt)
8, 9	Rückkehradresse bei Fehler
10	Fertigstellungskode
11, 12	spezifisch verwendet; kann manchmal auch entfallen bzw. enthält Adresse zu einem Zusatzparametervektor

Tab. 3 Parametervektorformat

DEVICE NAME	ENTRY POINT	MODULE ADDRESS	MAPPED UNITS
ZDOS	2600	2600 3FFF	0 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
CON	2100	2100 25FF	1 2 3
NULL	1D4F	0000 0000	20
PCON	0BEE	0000 0000	
FLOPPY	0BFD	0000 0000	

Tab. 4 Aktive Geräte nach der Systeminitialisierung

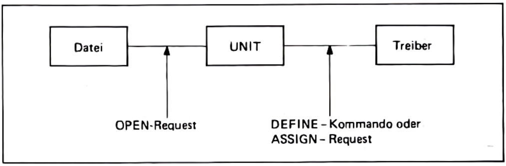


Abb. 5 Herstellen der Verbindung zwischen Datei und physischem Treiber

gig. Die meisten Requests von UDOS realisiert ZDOS, da es umfangreiche dateiorientierte E/A-Prozesse zu steuern hat. Bei konsoltypischen Geräten, die zeichenorientiert arbeiten (z. B. Tastatur, Bildschirm, Drucker, Lochband und ähnliches) ist der Umfang an Requests wesentlich geringer. Die am häufigsten benötigten Requests (nicht vollständig) werden bei den einzelnen Standardtreibern behandelt.

Das logische E/A-System von UDOS zeigt bezüglich seiner Funktionsweise damit gewisse Parallelen zu BDOS von SCPX (siehe 5.6.1.). Zusammengefaßt ergibt sich damit bei der E/A-Programmierung folgender Ablauf:

1. Es ist der für einen I/O-Request erforderliche Parametervektor im Vereinbarungsteil des Anwenderprogramms zu programmieren (auszufüllen). Wenn der betreffende Request einen Zusatzparametervektor erfordert (Byte 11,12 enthält dann dessen Anfangsadresse), so ist auch dieser zu vereinbaren.

2. Die Anfangsadresse des Parametervektors ist in das CPU-Register IY zu laden.

3. Es ist ein CALL 1003 auszuführen.

4. Nach Rückkehr aus dem E/A-System sollte das Byte 10 des Parametervektors abgetestet werden (Fehlerkode). Wird dort 80H zurückgemeldet, dann war die E/A-Operation erfolgreich. Alle anderen Kodierungen deuten auf E/A-Fehler oder Warnungen hin.

Sollen vom Anwender Treiber für eigene Hardware erstellt werden, so sind zunächst die E/A-Adressen für die Anwenderhardware festzulegen. Im A 5120 sind in der Konfiguration als Mikrorechnerentwicklungssystem nur E/A-Adressen kleiner als 80H belegt, das heißt, von 80H bis 0FFH sind 128 Adressen frei. Natürlich sind im Bürocomputer, aber bei anderen Konfigurationen, auch in diesem Bereich E/A-Adressen vergeben (z. B. Lochbandeinheit von 90H bis 97H), was gegebenenfalls berücksichtigt werden muß. Als nächster Schritt sind die in Abhängigkeit vom zu behandelnden Treiberprogramm erforderlichen Requestes zu bestimmen und zu programmieren. Schließlich ist der Treiber mit RET abzuschließen. Damit erhält er Unterprogrammstatus. Nach dem Übersetzungs- und Bindelauf muß mit

%SET SUBTYPE OF Treibername TO 1

der Treiberstatus im Kennsatz vermerkt werden. Über

%ACTIVATE $Treibername

wird der Treiber dann als OS-Bestandteil in das System eingebunden. Er ist nun über den Systemeintrittspunkt (1003H) jederzeit erreichbar.

Logischer Floppytreiber (ZDOS)

ZDOS realisiert die Dateiarbeit entsprechend Punkt 4.3.2. Mit einem Umfang von 6 KByte ist ZDOS der größte UDOS-Treiber. Standardmäßig ist ZDOS das Mastergerät. Damit werden alle Requests, die sich auf unvollständige Dateibezeichnungen beziehen, zu ZDOS geleitet. Ebenso fällt die Verwaltung der Diskettenbelegungsmaps unter die Regie von ZDOS. Mit dem INITIALIZE-Request werden diese von allen Disketten in den Arbeitsspeicher geladen, mit einem CLOSE-Request wird die zugehörige Map auf die Diskette geschrieben. Damit wird organisiert, daß im Arbeitsspeicher immer der aktuelle Belegungsstand für jeden Sektor aller eingelegten Disketten verfügbar ist. Mit diesen Informationen ist ZDOS nun in der Lage, bei Schreiboperationen die Records immer auf freie Sektoren der betreffenden Diskette zu plazieren.

Bei der Arbeit mit Dateien organisiert ZDOS die für UDOS typische Zeigerverwaltung. Wie bereits erläutert, sind an jedem Record zwei Zeiger angehängt, die die Adressen des nächsten bzw. des vorhergehenden Records markieren. ZDOS verwaltet nun insgesamt drei Zeiger:

– PREVIOUS RECORD POINTER (PRP)
vorhergehender Record

– CURRENT RECORD POINTER (CRP)
aktueller Record

– NEXT RECORD POINTER (NRP)
nachfolgender Record.

Bei jedem Zugriffs-Request werden diese Zeiger aktualisiert. Beim sequentiellen Lesen (READ BINARY) zum Beispiel wird der durch NRP adressierte Record einschließlich seiner Zeigerinformationen gelesen. Der bisherige CRP wird zum PRP. Die Angabe NRP wird aus der Zeigerinformation des gelesenen Records entnommen. Gleichfalls wird die Zeigerkette auf Widerspruchsfreiheit kontrolliert. Diesbezügliche Fehler werden als POINTER CHECK ERROR bezeichnet und sind zunächst zu reparieren, ehe mit der Datei weiter gearbeitet wird (z. B. mit dem Dienstprogramm FILE.DEBUG).

Nachfolgend sollen die ZDOS-Requests kurz skizziert werden. Die insgesamt 22 zu realisierenden Requests kann man in vier Gruppen zusammenfassen:

- *Standardrequests*

INITIALIZE
Einlesen aller Diskettenbelegungsmaps.

ASSIGN
Ordnet einer UNIT eine Datei zu.

OPEN
Öffnet eine Datei, liest DESCRIPTOR ein

CLOSE
Schließt eine Datei, aktualisiert DESCRIPTOR und Diskettenbelegungsmap auf der Diskette.

- *Positionierrequests*

REWIND
NRP zeigt auf den ersten Record.

SKIP FOREWARD
CRP wird um eine programmierte Anzahl von Records vorwärts bewegt.
SKIP BACKWARD
CRP wird um eine programmierte Anzahl von Records rückwärts bewegt.
SKIP TO END
CRP wird auf den letzten Record positioniert.

- *Lese- und Schreibrequests*

READ BINARY
Lesen sequentiell vorwärts.
WRITE BINARY
Schreiben sequentiell vorwärts.
READ CURRENT
Lesen des aktuellen Records.
WRITE CURRENT
Schreiben des aktuellen Records.
READ PREVIOUS
Lesen sequentiell rückwärts.
READ DIRECT
Lesen eines direkt adressierten Records, dessen Diskettenadresse im Zusatzparameterblock zu programmieren ist. (Achtung: Der Programmierer trägt für die Richtigkeit der Diskettenadresse selbst die Verantwortung!)
READ AND DELETE
Lesen des nächsten Records und Löschen der Daten (neue Daten können mit READ CURRENT zurückgeschrieben werden).

- *Requests zur Dateipflege*

DELETE
Löschen einer programmierten Anzahl von Records ab aktueller Recordposition.
DELETE REMAINING RECORDS
Löschen aller Records von CRP bis Dateiende.
ERASE
Alle Informationen der Datei im DIRECTORY werden gelöscht, DESCRIPTOR und Records werden in der Diskettenbelegungsmap freigegeben.
RENAME
Die Datei erhält einen neuen Namen.
UPDATE
Aktualisiert DESCRIPTOR und Map, schließt die Datei aber nicht.
SET ATTRIBUTES
DESCRIPTOR aktualisieren.
QUERY ATTRIBUTES
DESCRIPTOR im Anwenderprogramm bereitstellen.

Logischer Konsoltreiber CON

Der logische Konsoltreiber CON hebt die Arbeit mit den Geräten Tastatur, Bildschirm und Drucker gegenüber dem physischen Konsoltreiber auf eine neue Stufe. Mit den Requests werden stets Datensätze, also Records, bearbeitet, obwohl die konsoltypischen Geräte zeichenorientiert arbeiten. Damit werden diese Geräte auf das Niveau der standardisierten E/A-Schnittstelle (I/O-Vektor) gehoben.

Der Konsoltreiber CON ist Bestandteil der Executive OS. Seine Länge beträgt 500H Bytes. Er realisiert im System folgende Grundfunktionen:

1. Jedes von der Tastatur eingegebene Zeichen wird zum Bildschirm transportiert und auf der letzten Bildschirmzeile angezeigt.
2. Der Konsoltreiber bedient drei logische Geräte: die Konsoleingabe (1 = CONIN), die Konsolausgabe (2 = CONOUT) und das Listgerät (3 = SYSLST).
3. Es werden Bildschirmsteuerzeichen umgesetzt (Tab. 5) bzw. Konsolbetriebsarten eingestellt (Tab. 6).
4. Im Konsoltreiber wird eine Tabulatorzeile verwaltet (Länge 134 Positionen). Der Tabulator (CONTROL-I 09H) wird in eine entsprechende Anzahl von 'Space' umgewandelt, wenn er an die Konsole ausgegeben wird (CONTROL-Funktion über CI-Taste und Zeichentaste realisiert). Ein Tabulator kann an der Kursorposition durch folgende Zeichenfolge gesetzt werden:

'CTRL-T' 'T' (14H, 54H).
Das Löschen erfolgt mit
'CTRL-T' 'Space' (14H, 20H).

Steuerzeichen	Bezeichnung	Wirkung
08H	CHRDEL	letztes Zeichen im Eingabepuffer wird gelöscht; Kursor eine Position zurück
7FH	LINDEL	löscht im Eingabepuffer alle Zeichen, beginnend an der aktuellen Position rückwärts bis zum nächsten 'CR'
SAH	LF	Kursor auf Anfang der nächsten Zeile; 'Space' in den Eingabepuffer, kein 'CR'
0DH	CR	beendet Eingabe

Tab. 5 Bildschirmsteuerzeichen

Betriebsart	Wirkung
AUTOLF	bei jedem 'CR' wird automatisch ein 'LF' eingefügt (bei Betriebsart AUTOLF ON)
NULLCT	Angabe der Nullzeichen (OH), die an jedes 'CR' angehängt werden
LFCNT	Anzahl der 'LF'-Zeichen, die an jedes 'CR' angehängt werden bei Betriebsart AUTOLF ON
ECHO	jedes Zeichen wird nochmals an die Konsole zurückgesendet (in Betriebsart ECHO ON)

Tab. 6 Konsolbetriebsarten

Voreingestellt sind Tabulatoren aller acht Spalten, beginnend von links mit der Spalte 0. Diese Voreinstellung kann geändert werden mit den oben genannten Zeichenfolgen bzw. mit
%SET TABSIZE='n'
(Tabulatoren aller n Spalten).
Soll eine bestimmte Tabulatorbelegung gespeichert werden, so ist das mit dem Kommando SET_TABS möglich. Mit RESTORE_TABS kann sie in die Konsole zurückgeholt werden (siehe 4.5.3.).
Folgende I/O-Requests werden vom logischen Konsoltreiber verarbeitet:

INITIALIZE
Standardvoreinstellung wiederherstellen
ASSIGN
Nulloperation
OPEN
Nulloperation
CLOSE
Nulloperation
READ BINARY
Datenübertragung von der Konsole zum Speicher
WRITE BINARY
Datenübertragung vom Speicher zur Konsole, bis im Zeichenstrom 0FFH erscheint!
READ LINE
Datenübertragung von der Konsole zum Speicher

WRITE LINE
Datenübertragung vom Speicher zur Konsole, bis 'CR' im Zeichenstrom erscheint!
READ ABSOLUTE
Datenübertragung von der Konsole zum Speicher
WRITE ABSOLUTE
Datenübertragung vom Speicher zur Konsole entsprechend der im I/O-Vektor programmierten Datenbereichslänge (Byte 4,5)
READ STATUS
Lesen Konsolstatus
WRITE STATUS
Schreiben Konsolstatus
DEACTIVATE
Nulloperation

4.6.2.
Physischer Teil

Im physischen Teil von UDOS sind neben den physischen Gerätetreibern für Tastatur, Bildschirm und Floppy noch ein Systemdebugger und die Systemdaten untergebracht. Der gesamte physische Teil umfaßt die unteren 4 KByte (Adresse 0 bis 0FFH) des Arbeitsspeichers. Die einzelnen Komponenten sind in folgender Reihenfolge angeordnet:
- Systemdebugger
- physischer Konsoltreiber PCON
- physischer Floppytreiber FLOPPY
- Systemdaten.

Auf dieser Ebene ist eine direkte Manipulation des gesamten Systems möglich, das heißt also der oben genannten Geräte wie auch des Arbeitsspeichers. Da alle Kontroll- und Überwachungsmechanismen nur auf der logischen Ebene angesiedelt sind, stehen hier alle Möglichkeiten offen. Das ist insbesondere im Havariefall (Systemabsturz) ein großer Vorteil, da zum Beispiel mit den physischen Diskettenkommandos eventuell defekte Datenbestände wieder repariert werden können. Eine Nutzung dieser Ebene erfordert andererseits natürlich eine genaue Kenntnis des Gesamtsystems einschließlich der internen Abläufe sowie langjährige Erfahrung in der Systemprogrammierung, da bei Fehlbedienungen der Inhalt ganzer Disketten unbrauchbar werden kann. Besondere Vorsicht ist bei allen Schreiboperationen auf Diskette geboten.

Systemdebugger

In dieser Komponente stehen Kommandos für den Test von Maschinenprogrammen und Systeminformationen, wie zum Beispiel Speicher- und Registerinhalte, zur Verfügung. Alle diese Informationen sind änderbar. Auf dieser Ebene werden auch die physische Diskettenarbeit sowie Zugriffe zu den E/A-Ports erlaubt. Der Debugger meldet sich mit Promptzeichen der physischen Ebene '+' und erwartet eine Kommandoeingabe. Die Kommandos sind mit 'CR' abzuschließen.

Nachfolgend werden die wichtigsten Kommandos erläutert. Optionale Parameter stehen in eckigen Klammern. Sämtliche numerischen Eingaben und Ausgaben erfolgen hexadezimal. Von den Kommandos sind nur die Großbuchstaben einzugeben.

- *Registeranzeige und -manipulation*

+ Register [Registername]
Ohne Parameter werden die Inhalte sämtlicher CPU-Register incl. Alternativregistersatz und der 16-Bit-Register angezeigt. Mit einem Registernamen wird nur der Inhalt des einen spezifizierten Registers angezeigt. Es kann durch eine unmittelbar nachfolgende numerische Eingabe geändert werden.

- *Speicheranzeige und -manipulation*

+ Display [Adresse] [Länge]
Fehlt die Adresse, so wird diese intern auf Null gesetzt; fehlt die Längenangabe, so steht intern dafür eine Eins.
Mit diesem Kommando können einzelne Speicherzellen (Länge = 1) angezeigt und analog dem R-Kommando auch modifiziert bzw. ganze Speicherbereiche (Länge > 1) angezeigt werden. Im letzteren Fall werden alle Zeichen, die druckbar sind, noch zusätzlich in ASCII-Form angezeigt.

- *Speichertransport*

+ Move Zieladresse Quelladresse Länge
Hiermit wird ein Speicherbereich, der durch seine Anfangsadresse (Quelladresse) und Länge definiert ist, ab der Zieladresse erneut in den Speicher eingeschrieben.

- *Programmstart*

+ Jump [Adresse]
+ Go [Adresse]
Mit beiden Kommandos kann ein durch die Adresse angegebenes Programm gestartet werden, mit Jump unter Systemstatusbedingungen, mit Go unter Anwenderstatusbedingungen. Letzterer ist durch die eingestellten Werte in den Registern gegeben. Systemstatus ist gekennzeichnet durch:
Stackpointer SP = OCFFH Interruptmodus IM2
I-Register I = OFH Interrupt enable EI.
Jump sollte daher zum Abarbeiten getesteter Programme verwendet werden, Go zum Austesten von Programmen. Fehlt die optionale Adresse bei diesen Kommandos, dann wird die im PC-Register eingestellte verwendet.

- *Software-Breakpoint (Unterbrechungspunkt)*

+ Break [Adresse]
Fehlt die Adresse, dann wird das Breakpoint-Register (Software!) gelöscht, anderenfalls wird die angegebene Adresse als Breakpoint markiert. Wird bei Abarbeitung eines Programms dieser Breakpoint erreicht, dann wird das Programm an dieser Stelle abgebrochen, und es werden alle Register angezeigt. Mit Go läßt sich das Programm an der unterbrochenen Stelle fortsetzen.

- *Porteingabe und Portausgabe*

+ Port Input Portadresse
+ Port Output Portadresse Bitmuster
Mit diesen Kommandos kann man E/A-Ports byteweise direkt programmieren. Mit PI wird der adressierte Port abgefragt und das eingelesene Byte auf Konsole angezeigt. Mit PO läßt sich ein Byte (Steuer- oder Datenbyte) auf den adressierten Port ausgeben.

- *Interruptsteuerung*

+ Interrupt [Disable]
Kurzform: I [D]
Wird nur I eingegeben, werden nachfolgende Interrupts erlaubt, bei ‚I D' verboten, das heißt vom System nicht akzeptiert.

- *Physische Diskettenarbeit*

Load Diskettenadresse Speicheradresse Länge
Store Diskettenadresse Speicheradresse Länge
Alle Eingaben erfolgen hexadezimal. Die Diskettenadresse enthält Spurnummer (die ersten beiden Stellen), Sektornummer (die letzten beiden Stellen) und Laufwerksangabe. Das Laufwerk wird durch Hinzuaddieren einer Konstante

zur Sektornummer verschlüsselt, und zwar beträgt dieser Summand bei
Laufwerk 0 0H
Laufwerk 1 20H
Laufwerk 2 40H
Laufwerk 3 60H.
Die Länge darf nur Werte von ganzzahligen Vielfachen von 80H annehmen. Ebenso ist ein Überschreiten der Spurgrenze nicht möglich.
Beispiele:

+L 1605 4000 80
Lädt von Spur 16H des Laufwerkes 0 den Sektor 5 (80H Bytes) in den Arbeitsspeicher ab Adresse 4000H.
+L 40 6000 180
Lädt von Laufwerk 2 ab Spur 0, Sektor 0, 3 Sektoren (180H Bytes) in den Arbeitsspeicher ab Adresse 6000H.
+S 120 8000 300
Schreibt vom Arbeitsspeicher ab Adresse 8000H 6 Sektoren (300H Byte) auf Laufwerk 1, beginnend bei Spur 1, Sektor 0.
Achtung: Alle 6 Sektoren bekommen die gleichen Zeigerinformation, und zwar von dem Sektor, der zuletzt gelesen wurde. Deshalb vor jedem S-Kommando immer erst den Bereich lesen und dann schreiben! Ebenso muß die Länge mit der Recordlänge übereinstimmen bzw. kürzer sein. Man geht absolut sicher, wenn man sektorweise arbeitet (lesen, schreiben).

Auf der physischen Ebene sind noch zwei weitere Kommandos möglich, die aber nichts mit dem Debugger zu tun haben, und zwar:
+Organize
+Quit.
Organize startet den Lade- und Initialisierungsvorgang des logischen Systems. Ab Version 4.0 wird dieses Kommando beim Laden intern erzeugt. Quit dient der Rückkehr in das den Debugger aufrufende Programm. Da der Debugger Unterprogrammstatus besitzt, kann er von jedem beliebigen Programm aus aufgerufen werden (über einen CALL zum Debuggereintrittspunkt OBFAH).

Physische Treiber

Zum Grundbestand des physischen Systems gehören als Treiber der physische Konsoltreiber PCON und der physische Floppytreiber FLOPPY.

- *Konsoltreiber PCON*

In dieser Komponente sind die Einzelzeichentreiber für Tastatur, Bildschirm und Drucker realisiert. Bezüglich der UDOS-Versionen gibt es hier einige Unterschiede zu beachten. Bis zur Version 3.0 wurde durch PCON die gesamte verfügbare Konsolperipherie bedient (vgl. 4.1.). Mit Einführung der Tastatur K 7637 (serielle Tastatur) mußte das Konzept geändert werden, indem das Betriebssystem konfigurationsspezifisch generiert werden muß. Damit sind ab Version 4.0 in PCON nur noch die drei Einzelzeichentreiber implementiert, deren äquivalente Hardwarebaugruppen auch tatsächlich zur Anlage gehören, also zum Beispiel Tastatur K 7637, Monitor K 7222 und Drucker SD 1157 mit IFSS-Interface. Wird Hardware ausgetauscht, also zum Beispiel die Tastatur K 7636 verwendet, dann ist durch einen Generiervorgang auch das System, genauer PCON, zu ändern.
Hauptmerkmal von PCON ist das Rollen des Bildschirminhaltes. Beim Erreichen des Zeilenendes sowie bei jedem 'CR' erfolgt automatisch ein Hochrollen des gesamten Bildschirminhaltes um eine Zeile. Diese Rollfunktion kann ab Version 4.0 von UDOS auf eine generierbare Anzahl von Zeilen beschränkt werden. Dadurch ist es möglich, daß ein bestimmter Bildschirmbereich konstant bleibt, wodurch zum Beispiel für den Bediener wichtige Meldungen nicht aus dem Bild hinausrollen.
In PCON ist auch eine Hardcopyfunktion für den Drucker umgesetzt. Wird nämlich die Monitor-Taste betätigt, dann wird jedes zum Bildschirm transportierte Zeichen auch gleichzeitig zum Drucker transportiert. Damit wird der Bildschirminhalt parallel mitgedruckt. Die Monitor-Taste funktioniert getriggert, das heißt, wird sie erneut betätigt, so wird der Drucker wieder abgetrennt. Da diese Parallelschaltung von Bildschirm und Drucker im physischen Konsoltreiber implementiert ist, funktioniert sie auch bei Arbeit mit dem Systemdebugger, ohne daß das logische System überhaupt benötigt wird. Damit ist eine sehr effektive Fehlersuche im Havariefall bzw. bei Programmen möglich, weil alle Arbeitsschritte mit dem Debugger auch auf dem Drucker protokolliert werden können.
Die Einzelzeichentreiber bestehen im allgemeinen aus einer Reihe von Unterprogrammen, von denen einige allgemeinen Charakter haben. Dazu gehören unter anderem die in Tab. 7 aufgeführten Unterprogramme.

Name	Entry	Wirkung
INA	0BE8H	Test, ob ein Zeichen von der Tastatur eingegeben wurde und Übernahme ins A-Register (wenn kein Zeichen anliegt, dann wird das Z-Flag gesetzt)
OUTA	0BEBH	Ausgabe eines Zeichens vom A-Register an den Bildschirm oder Drucker
WRTXT	0BF1H	Ausgabe einer über (HL) adressierten Zeichenkette; (HL) gibt die Länge der Zeichenkette an, (HL+1) zeigt auf das erste Zeichen
COMIN	0BF7H	Ausgabe des Promptzeichens auf der Konsole und Warten auf Eingabe einer Zeichenkette, die mit CR abzuschließen ist. Die Zeichenkette gelangt in den Eingabepuffer INBU (0D8AH ... 0E89H).

Tab. 7 Unterprogramme des physischen Konsoltreiber PCON mit allgemeiner Bedeutung

- *Floppytreiber FLOPPY*

Im Floppytreiber wird die direkte Floppy-Disk-Ansteuerung realisiert. Bis einschließlich Version 3.0 waren die Floppytreiber für 8"- und 5,25"-Laufwerke getrennt, das heißt, bezüglich Floppy-Disk gab es bei UDOS zwei Systeme. Ab Version 4.0 wird die Arbeit durch einen einheitlichen Floppy-Disk-Treiber realisiert. Bei der Generierung wird nur noch festgelegt, nach welchem Modus jedes Laufwerk arbeiten soll. Diese Treiber setzen die beiden Debuggerkommandos L und S um (siehe 4.6.2.). Damit sind auch die dort fixierten Grenzen hier gültig.

4.7. Programmentwicklungssoftware

Bei der Entwicklung von Programmen auf Assemblerniveau gelten prinzipiell die gleichen Bedingungen wie auch bei den beiden anderen Betriebssystemen SIOS und SCP. Zur Erfassung von Quelltexten existiert ein Editor, die Übersetzung erfolgt mit einem Assembler, die einzelnen Programmoduln werden mit einem Binder zu einem abarbeitungsfähigen Programm verbunden und mittels eines Testsystems ausgetestet. Das prinzipielle Zusammenspiel zeigt Abb. 6. Nachfolgend soll nur noch auf Besonderheiten dieser Komponenten eingegangen werden.

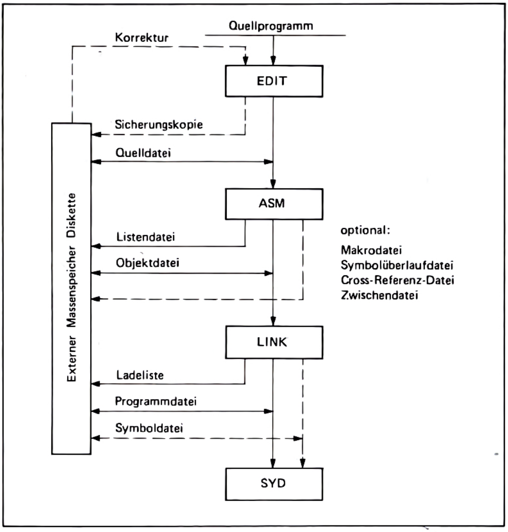


Abb.6 Prinzipieller Ablauf bei der Programmerstellung in Assemblertechnologie

4.7.1. Editor EDIT

Mit EDIT können Anwenderquelldateien erstellt und modifiziert werden. Der gesamte verfügbare Arbeitsspeicherbereich dient dem Editor als Arbeitspuffer. Damit kann ein relativ großes Quellprogramm (durchschnittlich > 1500 Quellzeilen) zu Editierzwecken im Speicher gehalten werden, wodurch die Ausführung der Editorkommandos stark beschleunigt wird. Wird die Größe des Editierpuffers überschritten, dann erfolgt der Datenaustausch von und zur Diskette automatisch. Man sollte aber aus Sicherheitsgründen nicht mit so großen Quellen arbeiten, sondern besser mit mehreren kleinen Teilprogrammen und diese dann getrennt übersetzen und mit dem Linker zusammenbinden.
Ein weiterer Vorzug dieses Editors besteht darin, daß er bei jedem Aufruf einer Quelldatei zunächst erst einmal eine Sicherheitskopie (sogenannte Back-up-Kopie) anlegt. Diese Sicherheitskopie erhält die Namenserweiterung '.OLD'.
Der Editor arbeitet zeilenorientiert. Innerhalb einer Zeile können Zeichenketten auf vielfältige Art und Weise modifiziert werden. Zur Bearbeitung einer Datei wird diese entsprechend der verfügbaren Speichergröße in den Arbeitsspeicher geladen. Mit den Editierkommandos kann sie jetzt bearbeitet werden. Zur Effektivierung des Editiervorganges besitzt der Editor einen Kommandopuffer, in dem ganze Kommandoketten zur Abarbeitung bereitgestellt und ausgeführt werden können. Weiterhin besteht die Möglichkeit, Teile der bearbeiteten Datei wiederum als Datei abzuspeichern oder andere Quelltexte in die gerade editierte Datei einzufügen.

4.7.2. Assembler ASM

Der Assembler selbst besteht aus den drei Dateien ASM, ASM2 und ASM3, die als Overlay-Dateien benutzt werden. ASM enthält das Wurzelsegment und den Teil für Pass 1. ASM2 wird zur Ausführung von Pass 2 benötigt und überlädt dazu einen Teil von ASM. Ist der optionale dritte Pass in der Kommandozeile spezifiziert, so wird zu dessen Ausführung ASM3 an die Stelle geladen, wo vorher ASM2 arbeitete. Mit diesem Mechanismus wird der vom Assembler benutzte Speicherraum relativ klein gehalten, was der Übersetzung größerer Quellprogramme zugute kommt.
Der UDOS-Assembler wandelt eine Quelldatei in eine verschiebliche Objektdatei um. Gleichzeitig wird eine Listendatei erstellt. Die Kennzeichnung erfolgt durch die Namenserweiterungen '.OBJ' und '.L'. Eine ganze Reihe von Optionen erlaubt es, den Übersetzungsvorgang zu modifizieren. So kann das Erzeugen der Objekt- und der Listendatei unterdrückt werden. Ebenso ist absoluter Objektkode möglich. Mittels eines dritten Passes kann eine sortierte Cross-Referenz-Liste für alle Marken mit den zugehörigen Adressen und Zeilennummern erstellt werden, die entweder an die Objekt- oder die Listendatei angehängt wird.
Der Assembler ist makrofähig und besitzt die Fähigkeit der bedingten Assemblierung.

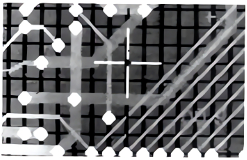
4.7.3. Programmverbinder LINK

Der Linker verarbeitet eine oder mehrere Objektdateien zu einer ausführbaren Programmdatei (Typ PROCEDURE). Die Programmdatei hat keine Namenserweiterung mehr. Der Linker löst die Adreßbeziehungen zwischen den separat erzeugten Objektdateien auf. Beim Linken können den Objektdateien feste Adressen zugewiesen werden, sie können aber auch hintereinander gebunden werden (indem man die Adreßangabe nur bei der ersten Objektdatei spezifiziert).
Der Linker ist als 2-Pass-System implementiert. Mittels eines optimalen dritten Passes kann eine Symboldatei mit der Namenserweiterung '.SYM' erstellt werden. Diese wird für die Programmtestung mit dem symbolischen Debugger SYD benötigt.
Mit Hilfe von Nur-Binde-Moduln, die lediglich zur Auflösung von Adreßbe-

ziehungen herangezogen, nicht aber selbst in den Programmkode eingebunden werden, lassen sich Überlagerungsstrukturen realisieren, wodurch große Programme speichereffektiv im Anwenderbereich eingeordnet werden können.

4.7.4. Symbolischer Debugger SYD

Während der Systemdebugger (siehe 4.6.2.) nur elementare Testfunktionen realisieren kann, steht mit dem symbolischen Debugger SYD eine komfortable Testhilfe zur Verfügung. Der wesentliche Qualitätsunterschied besteht in der symbolischen Fehlersuche, das heißt, es ist ein Zugriff zu Programm- und Datenstrukturen über die vom Programmierer festgelegten symbolischen Adressen möglich. Voraussetzung für eine derartige Arbeitsweise sind die Symbolinformationen, die in Form einer Symboldatei beim Linken erzeugt werden können.
Der symbolische Debugger arbeitet kommandoorientiert mit einem leistungsfähigen Kommandovorrat. Dazu gehören Kommandos zur Zustandsanzeige und -änderung, zur Programmlaufsteuerung und zur Systemverwaltung. Selbstverständlich sind alle Funktionen des Systemdebuggers auch nutzbar. SYD geht aber weit darüber hinaus; so sind zum Beispiel über SYD maximal 16 Breakpoints definierbar. Die Anzeige der Speicherinhalte kann hexadezimal oder rückassembliert (U-880-Mnemonik entsprechend TGL 26176) erfolgen. Mehrere Kommandos lassen sich zusammenfassen zu einer Kommandodatei und automatisch abarbeiten.

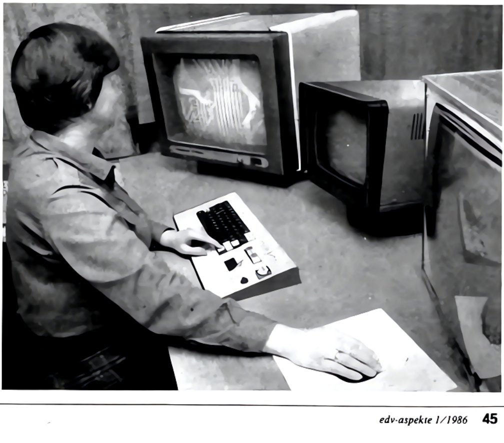


aspekte-Vorschau 2/86

Der Schlüsseltechnologie
CAD/CAM
ist aspekte 2/86 gewidmet.
Vorgestellt werden u.a. CAD/CAM-Lösungen in der metallverarbeitenden Industrie, im Bauwesen und im Bereich Elektrotechnik/Elektronik.
Aber auch Fragen der Aus- und Weiterbildung zu CAD/CAM werden behandelt.
Vorgesehen sind Artikel zu folgenden Komplexen:

- Durchgängige Lösungen
- Forschung und Entwicklung
- Entwurf, Konstruktion und Projektierung
- Technologische Vorbereitung
- Organisatorische Vorbereitung
- Produktionsdurchführung, einschließlich Versorgungs-, TUL-Prozesse und Materialwirtschaft
- Absatz und Vertrieb.

Bestellungen richten Sie bitte an den Verlag Die Wirtschaft, Abt. Vertrieb, 1055 Berlin, Am Friedrichshain 22.



Das Betriebssystem SCP

Hans-Gerd Unterschütz
VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt

5. SCP
5.1. Systemübersicht

Das Betriebssystem SCP 1520 ist ein diskettenorientiertes Einzelnutzer-Betriebssystem (SCP=Single User Control Program). Der Kern dieses Systems heißt SCPX 1526 und setzt einen Hauptspeicher von 64 Kbyte voraus. Es werden von SCPX eine Tastatur, ein Bildschirm, ein Drucker und ein bis vier Diskettenlaufwerke bedient. Dabei erfolgt der Aufruf dieser peripheren Geräte sequentiell, eine Simultanarbeit – wie z. B. bei SIOS 1520 – ist hier nicht möglich. SCPX realisiert die Anwenderschnittstelle des Betriebssystems CP/M, Version 2.2 (CP/M ist ein eingetragenes Warenzeichen der Firma Digital Research, Corp., USA), das heute wohl meistverbreiteten Betriebssystems für 8-Bit-Rechner.

Der Systemkern besteht aus drei Komponenten:
CCP – Console Command Processor
BDOS – Basic Disk Operating System
BIOS – Basic Input/Output System.

CCP ist ein Kommandointerpreter, der die vom Bediener eingegebenen Kommandos interpretiert und startet. Es werden residente und transiente Kommandos unterschieden. Während die Routinen der ersten in CCP selbst enthalten sind, werden die Routinen der transienten Kommandos zur Abarbeitung von einer Diskette in den Hauptspeicher eingelesen. Aber auch alle Programme, die sich als abarbeitbare Files auf den Disketten befinden, werden von CCP wie transiente Kommandos behandelt.

BDOS führt alle logischen Operationen zur Behandlung von Diskettendateien und der Ein- und Ausgabe zu den anderen peripheren Geräten aus. Diese Aufgabe wird in sogenannte BDOS-Funktionen unterteilt.

BIOS realisiert die physische Schnittstelle zu den einzelnen Geräten, also das Senden und Empfangen von Daten und Steuerinformationen. Am Beginn von BIOS ist eine Tabelle von Sprungbefehlen angelegt, die zu den einzelnen gerätespezifischen Treiberroutinen führen.

Damit ergibt sich eine einfache Struktur, die in Abb. 1 dargestellt ist.

Folgende periphere Geräte werden von SCPX bedient:

Tastatur:
– K 7637 – eine Tastatur mit seriellem Interface.
Besondere Bedeutung hat die Taste CTRL. Sie bewirkt eine bestimmte Veränderung des Codes einer gleichzeitig zu betätigenden Taste. Dies wird im folgenden durch einen Bindestrich zwischen den Tasten angedeutet, z. B. CTRL-C.
– K 7604, K 7606, K 7634, K 7636 – Der Anschluß dieser Tastaturen ist auch möglich. DO. hier die CTRL-Taste fehlt, wird deren Funktion durch das vorhergehende (!) Betätigen der Taste ET2 (bzw. Taste CNCL) realisiert.

Bildschirm:
– K 7222 – mit 1920 Zeichen (24 Zeilen mit je 80 Zeichen). Mit einigen Einschränkungen kann auch mit dem 1024-Zeichen-Bildschirm K 7221 (16 Zeilen mit je 64 Zeichen) gearbeitet werden.

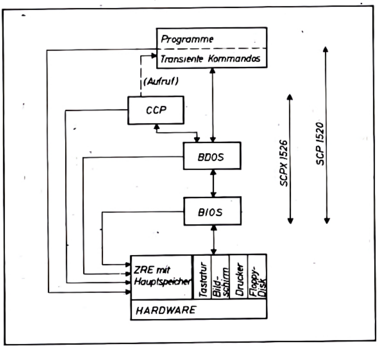
Drucker:
Es werden folgende Druckertypen vom SCPX unterstützt:
– SD 1152 mit Parallel- und seriellem Interface
– SD 1157 mit Parallel- und seriellem Interface
– K 6311, K 6312, K 6316.
Außerdem können auch Drucker mit V.24-Interface (DTR-Handshake) betrieben werden. Bei zweibahnigen Druckern wird die linke Walze von SCPX angesteuert.
Durch Ausgabe von Steuerfolgen in Anwenderprogrammen lassen sich sämtliche zugelassene Funktionen der genannten Drucker (z. B. andere Zeichenbreite, Druck auf rechter Walze usw.) realisieren.

Disketten-Laufwerke:
Es werden 1 bis maximal 4 Diskettenlaufwerke unterstützt. Folgende Typen von Disketten-Laufwerken sind betreibbar:
– MF 3200
– MF 6400
– K 5602
– K 5600.10
– K 5600.20.
Die Einheit von Diskette und Laufwerk wird als „logisches Laufwerk" und mit den Buchstaben A, B, C, D bezeichnet (im folgenden LW A, LW B usw.). Das erste Laufwerk mit Diskette ist das LW A, das zweite LW B, usw.

Bei SCP ist generell ein Hauptspeicher von 64 Kbyte erforderlich, es sind sowohl dynamische als auch statische RAM-Speicher zugelassen.

Die Generierung des SCPX für die entsprechende Gerätekonfiguration, das heißt das Erstellen der sogenannten Systemdiskette, wird vom Vertriebsbetrieb

Abb. 1 Strukturübersicht



ausgeführt. Mit dem transienten Kommando SYSG lassen sich beliebig viele weitere Systemdisketten kopieren.
An dieser Stelle ist aber zu erwähnen, daß neben den vom Systemkern bedienten Geräten auch weitere unter SCP betrieben werden können. So sind für die Anschlüsse von
– 1 bis 2 Kassettenmagnetbandgeräte K 5200
– $^1/_2$"-Magnetbandgerät CM 5300
– Zusatzdrucker (mit Anschluß IFSS)
– SLE
– 1 bis 4 HLE
Treiber entwickelt worden, die auf einfache Weise in die Anwenderprogramme in Unterprogramm (UP)-Technik einbindbar sind.
Der Operationsspeicher OSS (Typ 062-8471) mit 48 Kbyte läßt sich als zusätzlicher „externer" Datenspeicher nutzen. Für die Datenein- und -auslagerung ist das Unterprogramm ZSP48.REL zu nutzen.

5.2.
Systeminitialisierung
5.2.1.
Betriebsbeginn

Der erste Teil der Systeminitialisierung läuft ebenso ab wie bei SIOS und UDOS. Nach Einschalten der Anlage bzw. nach Auslösen der RESET-Funktion (zweimaliges Betätigen der Netz-Taste) wird zunächst das Lade-ROM-Programm aktiviert. Es liest von der Systemdiskette, die in das erste Laufwerk (also LW A) einzulegen ist, den *Systemlader* (SYL). Dieser belegt die ersten 512 Byte der Systemdiskette. Ist die Systemdiskette in einem anderen Laufwerk, so wird zwar das System auch eingelesen, es ist aber im Laufwerk A mindestens eine initialisierte Diskette erforderlich.
Nachdem der Systemlader in den Hauptspeicher eingelesen ist, meldet er sich nach Löschen des Bildschirmes mit „ROBOTRON LOADER".
Danach wird SCPX – ebenfalls von der Systemdiskette – in den Speicher gelesen und aufgerufen. Dies führt zu der Ausschrift
„SCPX 1526 – V. a.b (52k)".
a.b steht für die Versionsnummer.
Anschließend wird die Diskette im Laufwerk A angesprochen und die logische Verbindung zu ihr hergestellt. LW A wird das aktuelle logische Laufwerk, das sich in der Ausschrift
„A>"
auf dem Bildschirm dokumentiert. Das Zeichen > ist das Bereitschaftszeichen, das heißt, das System befindet sich jetzt im Grundzustand (CCP-Ebene), und es können vom Bediener Kommandos eingegeben werden. Alle Eingaben sind in Groß- und Kleinbuchstaben zugelassen. Durch CCP werden alle Kleinbuchstaben in Großbuchstaben gewandelt. Abzuschließen sind alle Eingaben mit der Taste ENTER bzw. Starttaste (außer ET2).
Tritt bei der Inbetriebnahme ein Fehler auf, so läßt sich aus den drei Ausschriften schließen, welcher Teil des Systems nicht in Ordnung ist. Kommt es zum Beispiel nur zu der Ausschrift „ROBOTRON LOADER", so fehlt auf der Systemdiskette entweder das SCPX, oder es ist nicht lesbar.

5.2.2.
Wahl des aktuellen Laufwerkes

Mit der Meldung „A>" zeigt das System an, daß das LW A das aktuelle Laufwerk ist. Das bedeutet: Alle Programm- und Dateiaufrufe beziehen sich auf das LW A, wenn eine Laufwerksangabe bei diesen Aufrufen fehlt. Durch die Eingabe „B:" läßt sich das LW B, durch „C:" das LW C usw. als aktuelles Laufwerk auswählen.
Beispiel: A>c:<ENTER>
C>_ (_=Kursor)
(In allen folgenden Beispielen werden <ENTER>, als Hinweis auf die Starttastenbetätigung am Ende einer Eingabe, und der Kursor nicht mehr aufgeführt.)
Im SCPX ist die Anzahl der logischen Laufwerke stets um eins größer als die Anzahl der physischen Laufwerke. Dem letzten physischen Laufwerk wird zusätzlich ein weiteres logisches Laufwerk mit der nächsten freien Bezeichnung zugeordnet. So realisiert zum Beispiel in einer Anlage mit drei physischen Laufwerken das dritte physische Laufwerk sowohl das logische Laufwerk C als auch das logische Laufwerk D. Dieses „zusätzliche" logische Laufwerk ist in der Lage, ein im Betriebssystem CP/M standardisiertes Diskettenformat (Datenaustauschformat) zu verarbeiten (s. 5.3.1.).

5.2.3.
Kaltstart, Warmstart

Als **Kaltstart** bezeichnet man den völligen Neubeginn der Arbeit mit SCP. Er wird hervorgerufen durch das Einschalten der Anlage oder durch die RESET-Funktion.
Ein **Warmstart** kann von einem Programm aufgerufen werden (z.B. am Programmende). Damit werden das System in den Grundzustand gebracht, logische Verbindungen zu vorher aktiven Laufwerken aufgehoben und das ursprüngliche aktuelle Laufwerk wieder eingestellt.
Wie aus der Struktur in Abb. 1 deutlich hervorgeht, ist im allgemeinen CCP für transiente Kommandos und Programme nur zum Aufruf erforderlich. Es können diese bei ihrer Abarbeitung den Speicherbereich von CCP zum Beispiel als Datenpuffer nutzen. Darum wird bei einem Warmstart in der Regel (Ausnahme: siehe residentes Kommando EXT) der Teil CCP des SCPX vom LW A (!) nachgeladen. Deshalb sollte im LW A stets eine Systemdiskette enthalten sein. Ein Warmstart läßt sich auch im Systemgrundzustand durch CTRL-C aufrufen. Dies ist zum Beispiel erforderlich, um in Laufwerken, auf die bereits zugegriffen wurde, einen Diskettenwechsel ausführen zu können.

5.3.
Disketten und Dateien
5.3.1.
Diskettenformate

Jede Diskette muß mit dem transienten Kommando INIT formatiert werden, bevor mit ihr gearbeitet werden kann. Aus einer Vielzahl möglicher Formate wurden für SCPX bezüglich Datenkapazität und Arbeitsgeschwindigkeit optimale Formate, im weiteren SCPX-Hausformate genannt, ausgewählt (Tab. 1).
Das zusätzliche logische Laufwerk er-

fordert ein Diskettenformat entsprechend Tab. 2.
Außerdem wird bei diesem Format ein Verschiebefaktor 6 realisiert. Das bedeutet: Auf der Diskette sind logisch aufeinanderfolgende Sektoren physisch nicht hintereinander aufgezeichnet, sondern zwischen ihnen liegen jeweils sechs Sektoren.
Mit Disketten dieses Formates ist der Daten- und Programmaustausch mit anderen CP/M-kompatiblen Betriebssystemen gewährleistet.

5.3.2.
Diskettenaufbau

Die Disketten dienen zum Speichern von Daten, Programmen und des Systems SCPX (einschließlich des Systemladers). Der Bereich für das SCPX wird stets freigehalten, so daß es sich empfiehlt, auf allen Disketten das System aufzubringen. Nur auf Disketten mit dem Daten-Austausch-Format kann kein System aufgezeichnet werden. Daten und Programme sind auf der Diskette als Dateien gespeichert. Alle Dateien sind in einem Dateiverzeichnis registriert.
Folgende Diskettenbelegung ergibt sich bei den einzelnen Formaten und Disketten:

- *SCPX (einschl. SYL):*
  – Spur 0–1
  bei 8"-Disketten (MFM-Verfahren)
  – Spur 0–2
  bei 8"-Disketten (FM-Verfahren) und 5,25"-Disketten
- *Dateiverzeichnis:*
  – Spur 2, Sektor 1–4
  bei 8"-Disketten (MFM-Verfahren)
  – Spur 3, Sektor 1–2
  bei 8"-Disketten (FM-Verfahren)
  – Spur 3, Sektor 1–8
  bei 5,25" Disketten
  – Spur 2, Sektor 1–16
  bei Disketten mit Daten-Austausch-Format
- *Dateien:*
  Sie belegen den Rest der Diskette (keine Reservespuren !).

Zur Zuordnung von Diskettenbereichen zu Dateien und zur dynamischen Verwaltung des Diskettenspeichers wird dieser ab dem Dateiverzeichnis in 2-Kbyte-Blöcke – bei Disketten mit Daten-Austausch-Format in 1-Kbyte-Blöcke – eingeteilt. Diese Blöcke werden gewissermaßen durchnumeriert, beginnend mit dem Block 0, der stets dem Dateiverzeichnis zugeordnet ist.

	8"-Diskette		5,25"-Diskette	
Aufzeichnungsverfahren	FM	MFM	MFM	MFM
Anzahl Spuren n	77	77	40	80
Anzahl Systemspuren	3	2	3	3
Spur 0: Sektoren/Spur	26	26(FM)	16	16
Byte/Sektor	128	128	256	256
Spur 1-(n-1): Sektoren/Spur	4	8	16	16
Byte/Sektor	1024	1024	256	256
Datenkapazität in KByte (ohne Dateiverzeichnis)	294	596	146	306

	8"-Diskette	5,25"-Diskette	
Aufzeichnungsverfahren	FM	MFM	MFM
Anzahl Spuren	77	40	80
Systemspuren	2	2	2
Sektoren/Spur	26	26	26
Byte/Sektor	128	128	128
Datenkapazität in KByte (ohne Dateiverzeichnis)	241	121	251

Tab. 1 SCPX-Disketten-Hausformate
Tab. 2 Diskettenformate für zusätzliches Laufwerk

5.3.3.
Diskettenarbeit im SCPX

Der Zugriff auf eine Datei durch ein Kommando oder ein Programm erfolgt stets recordweise (1 Record=128 Byte)! 128 Records (= 16 Kbyte) werden als ein Extent bezeichnet.
Um die Zahl der physischen Zugriffe zu verringern, realisiert SCPX mit einem internen 1-Kbyte-Puffer eine geblockte Arbeitsweise. Mit Ausnahme des Zugriffs auf Disketten mit Daten-Austausch-Format werden generell 1024-Byte-Blöcke aufgezeichnet bzw. gelesen, die Verwaltung dafür übernimmt das Disketten-E/A-System von SCPX.
Bei Lese- und Schreibfehlern werden vom System maximal acht Wiederholungen ausgeführt. Das Lesen von der Diskette wird mit der CRC-Kontrolle überwacht, das Aufzeichnen wird durch ein sofortiges Lesen (read-after-write) geprüft.

5.3.4.
Dateiverzeichnis

In dem Dateiverzeichnis einer Diskette sind alle Dateien dieser Diskette eingetragen. Jede Eintragung ist 32 Byte lang, ein Record nimmt somit vier Eintragungen auf.
Die Anzahl der möglichen Eintragungen je Diskette richtet sich nach deren Speicherkapazität:
Bei 8"-Disketten mit MFM-Verfahren sind 128 Eintragungen maximal zugelassen, das Dateiverzeichnis belegt hier also vier Kbyte (Block 0 und 1).
Bei allen anderen Diskettenformaten sind 64 Eintragungen möglich, so daß das Dateiverzeichnis zwei Kbyte belegt (Block 0, bei Disketten für das zusätzliche logische Laufwerk Block 0 und 1).
Inhaltlicher Aufbau einer Eintragung:
Byte 0: User
Mittels des residenten Kommandos User oder der BDOS-Funktion 32 läßt sich für jede Datei ein Nutzer angeben, der alleiniges Zugriffsrecht auf diese Datei hat. Es sind die Nutzer 0...1FH möglich, Standard ist der Nutzer 0.

Byte 1–8: Dateiname
Ist der Dateiname kürzer als acht Byte, so werden die restlichen Bytes mit 20H belegt.
Byte 9–11: Dateityp
Unbelegte Bytes werden ebenfalls mit 20H belegt.
Byte 12–14:
interne Kennbytes für BDOS
Byte 15: rc
Hier wird die Anzahl der Records eingetragen (record count).
Byte 16–31: Blocknummer
In diesen 16 Bytes werden die Blocknummern der Datei eingetragen. Ist die Blocknummer ein zweistelliger Wert, so kann eine Eintragung nur einen Extent aufnehmen (bei der 8"-Diskette mit MFM). Bei einstelliger Blocknummer (Anzahl der Blöcke <257) sind zwei Extents je Eintragung möglich.

Die Eintragung ist unbelegt, wenn im Byte 0 der Wert E5H steht!
Eine Datei kann mehrere Eintragungen belegen. In den Bits 0–4 des Bytes 12 und in den Bits 0–3 des Bytes 14 wird dafür die Nummer des bzw. der Extents dieser Eintragung vermerkt. Die Angabe in rc bezieht sich stets auf den hier angegebenen Extent und kann darum maximal 80H sein.

5.3.5.
Dateibezeichnung

Die Dateibezeichnung dient der Identifikation einer Diskettendatei in einem Kommando oder in einem Programm. Sie hat folgenden grundsätzlichen Aufbau:
d:dateiname.dateityp
Mit d wird das betreffende logische Laufwerk mit der zu lesenden oder aufzuzeichnenden Datei ausgewählt. Die Angabe ist nicht erforderlich, wenn es sich um eine Datei des aktuellen logischen Laufwerks handelt.
Der Dateiname besteht aus maximal acht Zeichen, der Dateityp aus maximal drei Zeichen. Dateiname und Dateityp werden durch einen Punkt getrennt. Es sind alle alphanumerischen Zeichen und Sonderzeichen bis auf folgende zugelassen:
<>;:=._*?
Das Leerzeichen (Code:20H) ist also auch ein zugelassenes Zeichen. Ebenso darf der Dateiname mit einer Ziffer beginnen. Ist der Name kürzer als acht oder der Typ kürzer als drei Zeichen, so werden bei den Kommandos die restlichen Stellen als Leerzeichen interpretiert, zum Beispiel bei der Eintragung in das Dateiverzeichnis.
Die Zeichen * und ? haben bei bestimmten Kommandos eine besondere Bedeutung, mit ihnen können Dateigruppen ausgewählt werden:
Ein Fragezeichen innerhalb eines Dateinamens oder eines Dateityps steht für jedes beliebige andere Zeichen an genau dieser Stelle. Ein Kommando bezieht sich dann auf alle Dateien, deren Bezeichnungen sich nur in den Positionen des Fragezeichens unterscheiden.
Beispiel: B:TEXT??.?NR
Es werden mit dieser Dateibezeichnung alle Dateien des logischen Laufwerks B angesprochen, deren Name aus maximal sechs Zeichen besteht und mit TEXT beginnt und deren Typ drei Zeichen lang ist und mit NR endet. Das könnten z. B. folgende Dateien sein: TEXT1.ANR , TEXT.1NR , TEXTXY./Nr .
Ein Stern(*) im Namen oder im Typ ersetzt alle noch folgenden Zeichen ab dieser Position.
Beispiel: A:FILE*.X*
Ein Kommando, das diese Dateibezeichnung verwendet, ruft alle Dateien des logischen Laufwerks A auf, deren Name mit FILE und deren Typ mit X beginnt; z. B. FILE1.XY , FILE0002.X, FILE.X+A .
Die Dateibezeichnung *.* steht für alle Dateien des aktuellen logischen Laufwerks.
Eine Kombination von Fragezeichen und Stern im Namen oder im Typ ist möglich, wobei ein Fragezeichen hinter dem Stern natürlich sinnlos ist.

5.3.6.
Dateityp

Der Dateityp innerhalb einer Dateibezeichnung dient im allgemeinen der Charakterisierung artgleicher Dateien. Bei zahlreichen Kommandos und Programmen werden bestimmte Dateitypen verlangt und/oder erzeugt. So gibt es u. a. folgende festgelegte Typen:
COM – für alle transienten Kommandos und Programme im Maschinencode des U 880 (Befehlsdatei)
MAC – für Assembler-Quellprogramme
BAK – für Sicherungsdateien
REL – für verschiebliche Objektcodedateien
SUB – für Textdateien mit Kommandos und Programmen, die durch das transiente Kommando SUB abgearbeitet werden können
□□□ – für Zwischendateien, die zur Sicherung oder in Erwartung einer neuen Datei angelegt werden.
Dies ist nur eine kleine Auswahl der festgelegten Dateitypen, weitere Typen sind den Beschreibungen der transienten Kommandos oder der Programme zu entnehmen.

5.3.7.
Dateiattribute

Jeder Datei werden Attribute zugeordnet; diese legen die Nutzung der Datei fest. Es gibt vier Attribute, jeweils zwei sind alternativ einstellbar:

R/W (read/write):
Dateien mit diesem Attribut können gelesen, überschrieben, erweitert oder gelöscht werden.
R/O (read only):
Dieses Attribut kennzeichnet eine schreibgeschützte Datei. Sie kann also nicht überschrieben, erweitert oder gelöscht werden. (Alternative zum R/W-Attribut)
DIR (Directory – Datei):
Dateien mit diesem Attribut werden durch das residente Kommando DIR auf dem Bildschirm angezeigt.
SYS (System – Datei):
Es erfolgt keine Anzeige der Dateien mit diesem Attribut beim Kommando DIR.
(Alternative zum DIR-Attribut)
Der Standardfall sind Dateien mit den Attributen R/W und DIR. Die Attribute einer Datei sind in den Bits 7 der Bytes 9 (R/W – R/O) und 10 (DIR-SYS) der Dateiverzeichniseintragung vermerkt. Sind die Bits gesetzt, so ist die Datei eine R/O- und SYS-Datei. Die Attribute einer Datei lassen sich mit dem transienten Kommando STAT oder – innerhalb eines Programms – mit der BDOS-Funktion 30 ändern.

5.3.8.
Verwaltung des Diskettenspeichers

Sobald auf eine Diskette zugegriffen wird (z. B. beim Warmstart auf das LW

A), wird von dieser Diskette der sogenannte Belegungsvektor im Hauptspeicher (BIOS) gebildet. In diesem Vektor ist jedem Diskettenblock ein Bit zugeordnet. Die Bildung des Vektors erfolgt durch das Setzen der Bits, die den Blocknummern in den belegten Dateiverzeichniseintragungen entsprechen.
Wird eine Datei um einen Block erweitert, so wird in der Umgebung des letzten Blockes dieser Datei nach einer freien Blocknummer (Bit im Belegungsvektor = 0) gesucht. Beim Löschen einer Datei werden die von ihr belegten Blocknummernbits im Belegungsvektor gelöscht, außerdem erhält das Byte 0 der Dateiverzeichniseintragung(en) den Wert E5H.
Jedes logische Laufwerk hat seinen eigenen Belegungsvektor.
Die ersten Bits des Vektors sind entsprechend der Dateiverzeichnisgröße stets gesetzt. Bei 8"-Disketten (MFM) und bei Disketten für das zusätzliche logische Laufwerk sind es Bit 0 und Bit 1, bei allen anderen Disketten nur das Bit 0 (Dateiverzeichnis belegt nur einen Block).

5.4.
Arbeitsspeicheraufteilung

Nach einem Kalt- oder Warmstart ergibt sich für den Hauptspeicher mit der Größe von 64 Kbyte die in Abb.2 dargestellte Aufteilung.
Die ersten 256 Byte (Adresse 0 ... FFH) sind für SCPX reserviert. Hier im Systemdatenbereich sind die wichtigsten Adressen, Kennbytes und Standardpuffer des Systems abgelegt bzw. eingerichtet.
Einige wichtige Speicherzellen in diesem Bereich und ihre Bedeutung:

00 – 02:
Sprungbefehl an die Warmstartroutine im Teil BIOS des SCPX. Diese Routine beginnt dort selbst mit einem Sprungbefehl.
04:
Angabe des aktuellen logischen Laufwerks:
für LW A ... 0
für LW B ... 1
für LW C ... 2
für LW D ... 3
für LW E ... 4
05 – 07:
Sprungbefehl an BDOS
38H – 3AH:
reserviert für das transiente Kommando DU (Debugger)
40H – 43H:
ab Version 1.1 belegt von der systeminternen Uhr.
Auf den Adressen 40H-42H steht die Zeit im BCD-Format in der Reihenfolge Stunde, Minute, Sekunde. Sie ist die vergangene Zeit seit dem letzten Kaltstart.
Durch ein Programm (oder durch den Debugger) kann eine beliebige andere Startzeit für die Systemuhr eingetragen werden.
55H – 7FH:
Bereich für den Standard-File-Control-Block (FCB)(s. 5.6.1.)
80H – FFH:
Standard-Ein-/Ausgabepuffer bei Diskettenarbeit.

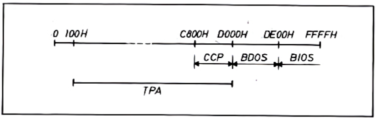


Abb. 2 Hauptspeicheraufteilung

Mit der Adresse 100 H beginnt der Programm- und Datenbereich für die transienten Kommandos und Programme. Er wird mit TPA (Transient Program Area) bezeichnet. Wie bereits erwähnt, kann im Normalfall (Ausnahme: siehe residentes Kommando EXT) der CCP-Bereich von Programmen überlagert werden, so daß maximal 52 Kbyte – entsprechend der Adresse 0D000H – dem Anwender zur Verfügung stehen. Diese Angabe steht deshalb auch in der Systemmeldung gleich hinter der Versionsnummer. (Genaugenommen sind die 256 Byte des Systemdatenbereiches davon noch abzuziehen.)
Wird in Programmen die Größe des zur Verfügung stehenden Speicherbereiches benötigt, so sollte stets der H-Teil der BDOS-Adresse auf der Adresse 07 zur Berechnung herangezogen werden. Gründe dafür sind zum einen die oben genannten Ausnahmen und zum anderen die Kompatibilität zu anderen Systemen.
Der Bildwiederholspeicher liegt wie bei allen Betriebssystemen des Bürocomputers A 5120/A 5130 am Ende des Hauptspeichers. Während der des 1920-Zeichen-Bildschirmes an der Adresse F800H beginnt, fängt der Bildwiederholspeicher des 1024-Zeichen-Bildschirmes bei FC00H an. Der 1-Kbyte-Bereich davor bleibt dann frei, er könnte im Bedarfsfall von Programmen genutzt werden.

5.5.
Das Kommandosystem
5.5.1.
Kommandoeingabe

Die Eingabe eines Kommandos und damit dessen Aufruf ist stets im Systemgrundzustand, der durch die Bereitschaftsmeldung
A>
oder B>, C>, D>, E> gekennzeichnet ist, möglich.
Wie bereits erwähnt, gibt es residente und transiente Kommandos. Die residenten Kommandos sind Bestandteil des CCP, während die transienten Kommandos als Dateien vom Typ COM auf der Diskette vorliegen und zu ihrer Abarbeitung in den Hauptspeicher eingelesen werden. Beim Aufruf eines transienten Kommandos darf nur der Name und nicht der Typ COM eingegeben werden.
Nach dem Namen des Kommandos können sich ein oder mehrere Parameter anschließen, die durch Leerschritte untereinander und vom Kommandonamen getrennt sind.
Bei der Übernahme der Zeichen in den CCP-Puffer werden alle Kleinbuchstaben in Großbuchstaben gewandelt (der Code 61H ... 7AH in 41H ... 5AH)!
Bei der Eingabe der Kommandozeile ermöglicht CCP durch Steuertasten verschiedene Zeilen-Editier-Funktionen:

CTRL-H ...
Kursor um ein Zeichen nach links und

Löschen des Zeichens an dieser Stelle
CTRL-X ...
Löschen der gesamten Eingabezeile auf dem Bildschirm und im CCP-Puffer; Kursor zur Anfangsposition
CTRL-E ...
Kursor auf neue Zeile (kein Ende der Eingabe!)
CTRL-U ...
Löschen der gesamten Eingabezeile im CCP-Puffer, die Zeichen bleiben aber auf dem Bildschirm erhalten. Der Kursor wird zur Neueingabe auf die nächste Zeile positioniert.
DEL ...
Es wird das letzte Zeichen im CCP-Puffer gelöscht. Auf dem Bildschirm erscheint das gelöschte Zeichen noch einmal!
CTRL-R ...
nochmaliges Anzeigen des CCP-Puffer-Inhaltes auf der nächsten Zeile (sinnvoll nach mehrfachem DEL).

Außerdem sind noch folgende Steuerfunktionen während der Kommandoeingabe möglich:

CTRL-C ...
Wird dieses Steuerzeichen an erster Stelle eingegeben, so wird ein Warmstart ausgelöst. Dies ist immer dann erforderlich, wenn zwischen zwei Kommandos ein Diskettenwechsel erfolgte und auf die neue Diskette schreibend zugegriffen werden soll. Das System hebt mit einem Warmstart alle logischen Laufwerke in den read/write-Status.
Ein CTRL-C an anderer Stelle wirkt nur wie die Eingabe des Codes 03H, auf dem Bildschirm erscheint: ∧C .
CTRL-P ...
Ab dieser Eingabe werden alle nachfolgend auf den Bildschirm ausgegebenen Zeichen auch zum Drucker übertragen. Es ist aber darauf zu achten, daß nicht alle Bildschirmsteuerzeichen auch vom Drucker akzeptiert werden.
Mit einem zweiten CTRL-P wird das Kopieren auf dem Drucker wieder abgebrochen.

Die Kommandoeingabe wird mit einer Starttaste, CTRL-J oder CTRL-M beendet.

5.5.2. Residente Kommandos

DIR

Mit dem Kommando DIR können Dateibezeichnungen aus dem Dateiverzeichnis eines logischen Laufwerkes auf dem Bildschirm angezeigt werden. Der Name DIR leitet sich von Directory, der englischen Bezeichnung für Dateiverzeichnis, ab. Das Format des Kommandos ist:
DIR[d:][dateibezeichnung] .
Als Parameter sind somit ein beliebiges logisches Laufwerk und ein- oder mehrdeutige Dateibezeichnungen zugelassen. Wird kein Parameter angegeben, so werden die Dateien des aktuellen logischen Laufwerkes auf dem Bildschirm angezeigt. Nicht dargestellt werden die Dateien mit dem Dateiattribut SYS!
Befindet sich die gesuchte Datei nicht auf der Diskette, so erscheint die Meldung
„NO FILE."
Beispiele:

dir b:test.*
Alle Dateien des LW B, deren Dateiname TEST lautet, werden auf dem Bildschirm aufgelistet.
dir datei1.mac
Es wird nach der Datei DATEI1.MAC auf dem aktuellen logischen Laufwerk gesucht. Ist sie vorhanden, so wird die Bezeichnung angezeigt.
dir c:
Alle Dateien des logischen Laufwerks C werden aufgelistet (identisch zu dir c:*.*).

Hier sei nochmals auf die Wirkung der Steuertaste CTRL-P hingewiesen, die eine Kopie der Bildschirmausgabe auf dem Drucker auslöst.

ERA

Mit diesem Kommando können Dateien gelöscht (erase) werden. Es sind stets Parameter erforderlich zur Angabe der zu löschenden Dateien. Das Kommandoformat ist:
ERA[d:]dateibezeichnung.
Bei der Dateibezeichnung sind sowohl eindeutige als auch mehrdeutige möglich. Es kann ein anderes Laufwerk als das aktuelle logische Laufwerk angewählt werden.
Beispiele:

era test.com
Die Datei TEST.COM auf dem aktuellen logischen Laufwerk wird gelöscht.
era c:*.dat
Alle Dateien des logischen Laufwerks C mit dem Dateityp DAT werden gelöscht.

Wird die mehrdeutige Dateibezeichnung *.* verwendet, so erscheint auf dem Bildschirm die Frage „ALL(Y/N)?", um ein Löschen aller Dateien durch eine versehentliche Eingabe zu verhindern. Nur bei der Eingabe von Y werden alle Dateien des adressierten logischen Laufwerks gelöscht.
Existiert eine Datei unter der angegebenen Bezeichnung nicht, so wird
„NO FILE"
gemeldet.
Während auch SYS-Dateien gelöscht werden können, ist das natürlich für R/O-Dateien nicht möglich. In diesem Fall kommt es zur Meldung
„SCPX ERROR ON d: FILE R/O".

REN

Mit diesem Kommando können Dateien umbenannt werden (rename).
Als Parameter ist zuerst die neue Dateibezeichnung einzugeben, danach folgt ein Gleichheitszeichen und die alte Dateibezeichnung:
REN (neue)dateibezeichnung=(alte)dateibezeichnung .
Eine Laufwerksangabe ist möglich, sie muß aber bei beiden Dateibezeichnungen identisch oder nur bei der neuen angegeben sein. Die Dateibezeichnungen dürfen kein ? und kein * enthalten, müssen also eindeutig sein.
Beispiel:

ren c:data.neu=daten.alt
Die Datei DATEN.ALT im LW C wird umbenannt in die Datei DATA.NEU.

Ist die umzubenennende Datei nicht auf dem ausgewählten logischen Laufwerk, so kommt es zu der SCPX-Meldung „NO FILE".
Gibt es bereits eine Datei mit der neuen Bezeichnung, so erfolgt keine Umbenennung, und SCPX meldet: „FILE EXISTS".
Bei der Umbenennung von SYS-Dateien entstehen DIR-Dateien! Der Versuch, R/O-Dateien umzubenennen, wird mit der Meldung
„SCPX ERROR ON d: FILE R/O"
beantwortet.

TYPE

Mit dem Kommando TYPE können Textdateien auf dem Bildschirm und – nach CTRL-P – auch auf dem Drucker

ausgegeben werden. Das Eingabeformat ist:
TYPE[d:]dateibezeichnung.
Die Dateibezeichnung muß eindeutig sein.
Beispiel:

type a:text.prn
Die Textdatei TEXT.PRN des logischen Laufwerks A wird auf dem Bildschirm rollend ausgegeben. Wurde CTRL-P vor der Starttaste eingegeben, erfolgt auch eine Ausgabe auf dem Drucker.

Mit der Steuertaste CTRL-S kann die Ausgabe angehalten und – mit einer zweiten Betätigung – wieder fortgesetzt werden.
Mit einer beliebigen anderen Taste wird die Ausgabe abgebrochen.

SAVE

Mit diesem Kommando können die Daten vom Hauptspeicher ab der Adresse 100H aufwärts auf einer Diskette unter einer beliebigen Dateibezeichnung gespeichert werden. Nach dem Kommandonamen SAVE sind stets zwei Parameter anzugeben:
– die Anzahl n der 256-Byte-Segmente (Angabe als Dezimalwert), die übertragen werden sollen und
– eine eindeutige Dateibezeichnung.
Somit ergibt sich folgendes Format:
SAVE n [d:]dateibezeichnung.
Beispiel:

save 4 b:infor.dat
Auf dem logischen Laufwerk B wird die Datei INFOR.DAT angelegt. Ihre Daten sind eine Kopie des Hauptspeicherbereichs 100H – 4FFH.

Sollte unter der gleichen Dateibezeichnung auf der Diskette bereits eine Datei existieren, so wird diese vor dem Aufzeichnen der neuen Datei gelöscht. Nur Dateien mit dem R/O-Attribut werden nicht gelöscht, es kommt zu den gleichen Meldungen wie bei ERA.
Die neuen Dateien haben die Attribute DIR und R/W.

USER

Mit diesem Kommando wird für alle folgenden Arbeiten mit Diskettendateien der Nutzer festgelegt. Das Format des Kommandos ist
USER n,
wobei n den Nutzer bezeichnet und Werte von 0 bis 15 annehmen kann. Der User 0 ist der Standardwert, er wird beim Kaltstart eingestellt.
Beispiel:

user 4
Es ist nun der Nutzer 4 eingestellt. Das bedeutet, daß nur noch auf Dateien des Nutzers 4 zugegriffen werden kann oder nur Dateien dieses Nutzers erstellt werden können. Für die residenten Kommandos bedeutet das z. B.:
– DIR zeigt nur Dateien des Nutzers 4 an
– ERA löscht nur Dateien des Nutzers 4
– SAVE erzeugt nur Dateien des Nutzers 4
usw.
Es können auch jetzt nur transiente Kommandos oder Programme (COM-Dateien) aufgerufen werden, in deren Dateieintragung der Nutzer 4 (auf dem Byte 0) registriert ist!

Man sollte im allgemeinen den Standard (Nutzer 0) verwenden, insbesondere für transiente Kommandos und Programme, um die Arbeit mit diesen zu vereinfachen; z. B. Aufruf, Kopieren. Außerdem wird die Diskettenkapazität nicht mit der mehrfachen Abspeicherung gleicher Programme oder Kommandos unter verschiedenen Nutzern unnötig reduziert.
Um Dateien von einem anderen Nutzer auf den aktuellen Nutzer zu kopieren, ist das transiente Kommando PIP mit dem Parameter Gn vorgesehen (siehe S. 55).

EXT

Dieses residente Kommando ist ab der Version 1.1 realisiert und ermöglicht es, die Zahl der residenten, das heißt ständig im Hauptspeicher verfügbaren Kommandos durch den Anwender zu erweitern (extend). Es können zusätzlich bis zu fünf beliebige transiente Kommandos oder Programme in residente Kommandos gewandelt werden. Zu diesem Zweck werden diese durch EXT im Hauptspeicher vor CCP angelegt. Bei später folgenden Aufrufen dieser nunmehr residenten Kommandos werden sie dann nur noch auf die Adresse 100H umgelagert, so daß dies wesentlich schneller geht als der sonst erforderliche Diskettenzugriff. Bei häufig genutzten Kommandos oder Programmen ist das sehr günstig. Außerdem besteht der Vorteil, daß diese auch nach Wechsel der Diskette, auf der sie gespeichert waren, verfügbar bleiben.
Das Format des Kommandos ist:
EXT [Ad:] dateibezeichnung
Bei der Dateibezeichnung muß nicht der Typ COM angegeben werden. Vom Dateinamen werden nur die ersten vier Zeichen zur Bezeichnung des zusätzlichen neuen residenten Kommandos übernommen, sie dürfen somit nicht mit den Namen der ursprünglichen residenten Kommandos übereinstimmen. (Ist das der Fall, so ist mit REN eine Umbenennung vor EXT vorzunehmen.)
Durch die Kommandoerweiterung verringert sich der TPA entsprechend. Außerdem kann CCP jetzt nicht mehr von einem Programm überschrieben werden, so daß nun ein Warmstart zu keinem Nachladen von CCP führt. Das kann ein positiver Nebeneffekt sein, da jetzt im LW A keine Systemdiskette erforderlich ist. Zur Berechnung des TPA kann weiterhin der H-Teil der BDOS-Adresse benutzt werden, da vor den eingebundenen zusätzlichen Kommandos ein Sprungbefehl an BDOS gesetzt wird und dieser nun als Adresse des Sprungbefehls auf den Adressen 5–7 dient.
Folgende Kontrollen und *Maßnahmen zur Speicherbereichüberwachung* werden ausgeführt:

- Bei Überschreitung der möglichen Anzahl neuer residenter Kommandos wird das Kommando EXT... ohne Meldung ignoriert.
- Beim Einfügen der neuen Kommandos vor CCP wird auf Unterschreitung der Adressse 200H geprüft. Ist dies der Fall, wird „BAD LOAD“ gemeldet, das residente Kommando RES (S. 53) intern ausgeführt und das Kommando EXT abgebrochen.
- Die gleiche Wirkung wird beim Aufruf eines transienten Kommandos oder eines Programms erzielt, wenn beim Laden (von der Diskette!) die nun verringerte TPA-Grenze erreicht wird.
- Wird ein neues residentes Kommando aufgerufen, wird ebenfalls die TPA-Grenze überwacht. Wird sie durch die Umlagerung überschritten, wird zwar das Kommando nochmals als residentes Kommando abgearbeitet, jedoch

wird vor der Ausführung das Kommando RES intern ausgeführt!

RES

Mit diesem Kommando wird wieder der Zustand des SCPX wie vor dem ersten EXT-Kommando hergestellt, alle neuen residenten Kommandos werden zurückgesetzt (reset). Damit wirken nur noch die ursprünglichen residenten Kommandos DIR, ERA, TYPE, REN, SAVE, EXT, RES und USER. Der TPA reicht wieder bis zum Anfang von BDOS.

5.5.3. Transiente Kommandos

Die transienten Kommandos werden als Dateien vom Typ COM auf der Systemdiskette ausgeliefert, da sie zum System SCP gehören. Sie unterscheiden sich sonst nicht von anderen Programmen. Natürlich lassen sie sich auf jede andere Diskette kopieren. Ihr Aufruf erfolgt im Systemgrundzustand durch die Angabe des Dateinamens. Die Auswahl eines beliebigen logischen Laufwerks vor dem Namen (mit :) ist möglich, so daß das aktuelle logische Laufwerk nicht umgestellt werden muß. Das allgemeine Format eines Aufrufs eines transienten Kommandos ist somit:

[d:] dateiname.

INIT

Das transiente Kommando INIT dient zum Formatieren von Disketten unter Steuerung von SCPX. Es werden folgende Normen für softsektorierte Disketten berücksichtigt:

KROS 5108, KROS 5110/01, ISO 5654, IBM 3740, ISO/TC 97/SC 11 Nr. 347 bzw. 209, IBM 34.

Es werden die Normen nur in bezug auf den physischen Spuraufbau, das heißt auf das Aufzeichnungsverfahren, den ID-Feldaufbau, die Lücken und Marken, beachtet.

Das Format des Kommandos ist INIT.

INIT formatiert die unter 5.3.1. erläuterten SCPX-Hausformate entsprechend der verwendeten Disketten- und Laufwerkstypen und das Datenaustauschformat. Zusätzlich ist INIT schon vorbereitet auf zweiseitige Diskettenlaufwerke.

Der Ablauf von INIT erfolgt bedienergeführt:

Zunächst kann der Bediener das logische Laufwerk angeben, auf dem eine Diskette formatiert werden soll. INIT bietet nun alle Formate an, die auf diesem Laufwerkstyp möglich sind. Nach der Auswahl durch den Bediener und dem Hinweis, daß nun die gesamte Diskette gelöscht wird, erfolgt das Formatieren. Die Spur, die gerade formatiert wird, wird angezeigt.

Am Ende des Formatierens werden — falls vorhanden — die defekten Spuren angezeigt. Über diese kann durch Bedienerauswahl eine Sperrdatei angelegt werden. In diesem Falle werden alle Blöcke der defekten Spuren zu einer Datei mit dem Nutzer 17 zusammengefaßt. Dieser Nutzer kann vom residenten Kommando USER nicht eingestellt werden! Die Diskettenkapazität sinkt dadurch, aber sonst merkt der Nutzer dieser Diskette nichts von den defekten Spuren. Ein Anlegen einer Sperrdatei ist natürlich nicht möglich, wenn die Spur mit dem Dateiverzeichnis defekt ist. Diese Diskette ist für SCP nicht verwendbar. Bei defekten Spuren im Bereich der Systemspuren ist eine Verwendung bei SCP möglich, diese Diskette kann nur nicht als Systemdiskette benutzt werden. Auftretende Schreib-/Leserfehler und Fehler der Hardware werden angezeigt und führen zum Abbruch des Formatiervorganges.

SYSG

Mit SYSG kann man das System SCPX kopieren oder generieren. Es bestehen folgende Möglichkeiten:

a) Es lassen sich die Systemspuren von Diskette zu Diskette gleichen Typs kopieren.

b) Das SCPX kann von den Systemspuren einer beliebigen Diskette in den Hauptspeicher eingelesen werden. Das System ist dann wie folgt im Speicher abgelegt:

ab 0780H	Systemlader SYL
ab 0980H	CCP
ab 1180H	BDOS
ab 1F80H	BIOS.

c) Ein sich im Hauptspeicher ab der Adressse 780H befindliches System einschließlich Systemlader kann auf die Systemspuren einer Diskette aufgezeichnet werden.

d) Es kann auch eine Diskettendatei als Systemquelle verwendet werden.

Der Fall a) ist der wichtigste und gebräuchlichste.

Das Kommando SYSG erlaubt es auch, daß nur mit einem Laufwerk Systemkopien hergestellt werden können.

Die Bedienerführung ist einfach und läßt ein mehrfaches Kopieren zu.

Für den Fall d) ist die Dateibezeichnung als Paramter, durch ein Leerzeichen von SYSG getrennt, anzugeben.

SEPR

Mit diesem transienten Kommando ist es möglich, das vom System unterstützte Drucker-Interface zu ändern. Das Format ist SEPR.

Es gibt keine Parameterangaben.

Nachdem die Datei SEPR.COM durch den Aufruf in den Speicher gelesen und gestartet wurde, kann der Bediener entscheiden, ob das Interface nur temporär — bis zum nächsten Kaltstart — im Hauptspeicher (BIOS) oder im System, das auf einer Diskette steht, geändert werden soll.

Folgende Schnittstellen sind auswählbar:

– Parallelschnittstelle für SD 1152 und SD 1157

– Serielle Schnittstelle (IFSS) für SD 1152, SD 1157, K 6311, K 6312 und K 6316

– V.24 mit den Baudraten 75 baud...9600 baud.

STAT

Die wichtigsten Informationen über Dateien und Disketten können durch STAT abgefragt werden. Weiterhin ist es möglich, den Status von Disketten und die Dateiattribute zu ändern.

Das Grundformat des Kommandos ist:

STAT [parameter]

Weitere Eingaben sind nicht erforderlich, so daß das Kommando nach der Anzeige der angeforderten Informationen oder einer Fehlerausschrift sofort wieder — mit einem Warmstart — in den Systemgrundzustand geht.

Folgende spezielle Formate sind möglich:

a) STAT

Es wird der noch freie Speicherplatz (in Kbyte) der Disketten des aktuellen logischen Laufwerks und der logischen Laufwerke, mit denen schon gearbeitet wurde, angezeigt. Außerdem wird der

Diskettenstatus (R/W oder R/O) angezeigt. Nach einem Warmstart sind alle Disketten im R/W-Status; bei einem Diskettenwechsel wird die neue Diskette in den R/O-Status gesetzt.
Beispiel:
A>stat führt zur Anzeige
A:R/W Space:140 k
b) STAT d:
Ohne Anzeige des Diskettenstatus wird der freie Speicherplatz der mit d ausgewählten Diskette angezeigt.
Beispiel:
C>stat b: führt zur Anzeige
BYTES REMAINING ON B: 248 k
c) STAT [d:] dateibezeichnung
Alle der Dateibezeichnung entsprechenden Dateien werden mit der Anzahl ihrer Records, der belegten Blöcke, der belegten Eintragungen und dem Attribut angezeigt. Hier ist die Möglichkeit der mehrdeutigen Dateibezeichnung mit Hilfe von ? und * besonders nützlich. Außerdem wird noch der freie Diskettenspeicherplatz wie bei b) angezeigt.
Beispiel:

```
A>stat *. com führt zur Anzeige
Recs   Bytes   Ext   Acc
54     8k      1     R/W A:TEST.COM
29     4k      1     R/O A:(PRUEF.COM)
Bytes Remaining On A: 258k
```

Wie das Beispiel zeigt, werden die Anzahl der Records, die belegte Speicherkapazität, die Anzahl der Einträge, das R/W- oder R/O-Attribut und die Dateibezeichnung angegebenen. Stehen der Name und Typ in Klammern, so ist die Datei eine SYS-Datei.
d) STAT [d:] dateibezeichnung ▫s
Mit diesem Format wird die Anzeige von Format c) um die Angabe Size erweitert. Bei sequentiell erstellten Dateien ist der bei Size angegebene Wert gleich der Recordangabe. Bei Direktzugriffsdateien können auch Löcher enthalten sein.
In diesem Falle zeigt Size die Anzahl von belegten und unbelegten Records an (entsprechend dem letzten Record dieser Datei), während Recs die Summe der Records der einzelnen Eintragungen enthält.
e) STAT [d:]=R/O
Es kann eine Diskette in den R/O-Status – bis zum nächsten Warmstart – versetzt werden. So kann zum Beispiel danach keine Datei dieser Diskette, auch wenn diese Datei das R/W-Attribut hat, gelöscht werden.
f) STAT [d]: dateibezeichnung ▫attribut
Mit diesem Format kann jeder Datei ein beliebiges Attribut gegeben werden. Für „attribut" ist r/w, r/o, dir oder sys zu setzen.
Beispiel:

A >stat init. com▫sys führt zur Anzeige
INIT.COM set to SYS
g) STAT [d:]DSK:
Hiermit werden zahlreiche Informationen über das Diskettenformat und den -aufbau aufgelistet. Fehlt d:, so werden diese Informationen von allen bereits aktivierten logischen Laufwerken angezeigt.
Folgende Angaben erscheinen auf dem Bildschirm:
– Laufwerksname (A, B, C, D oder E)
– Speicherkapazität in Anzahl Records
– Speicherkapazität in Kbyte
– Anzahl maximal möglicher Dateieintragungen (64 oder 128)
– Anzahl der Records je Eintragung (128 oder 256)
– Anzahl der Records je Block (8 oder 16)
– Anzahl der Records je Spur (26, 32 oder 64)
– Anzahl der reservierten Spuren für das System (2 oder 3)
h) STAT USR:
Damit werden der aktuelle Nutzer und alle Nutzer der Dateien, die auf der Diskette des aktuellen logischen Laufwerks stehen, angezeigt.
Beispiel:

```
C>stat usr: führt zur Anzeige
Active User : 0
Active Files : 0 1 4
```

SDIR
Dieses transiente Kommando stellt eine Erweiterung des residenten Kommandos DIR dar. Es kann der Inhalt von Disketten aller logischen Laufwerke der vorliegenden Gerätekonfiguration zur Anzeige gebracht werden.
Die Formate des Kommandos sind sehr vielfältig, die wichtigsten sind:
SDIR [d:][dateibezeichnung]
SDIR [option][d:]
SDIR [option][dateibezeichnung].
Die Optionen dienen der Auswahl von Ausgabeformaten, Dateiattributen, Nutzern und/oder logischen Laufwerken. Sie sind in eckige Klammern zu setzen und durch Komma oder Leerschritt untereinander zu trennen.
Der Disketteninhalt kann in drei Formaten dargestellt werden: FULL, SIZE, SHORT.
Mit FULL werden die meisten, mit SHORT die wenigsten Informationen (ähnlich DIR) zur Anzeige gebracht. Fehlt die Formatangabe, so wird FULL angenommen.
Bei FULL und SIZE wird neben den Informationen zu den einzelnen Dateien auch eine Summenzeile ausgegeben. Diese zeigt die durch die ausgewählten Dateien belegte Speicherkapazität, deren gesamte Anzahl Records, die Anzahl der aufgeführten Dateien selbst und die Anzahl möglicher und belegter Dateieintragungen an.
Die gewünschten Dateiattribute sind mit RW, RO, DIR und SYS als Option anzugeben.
Die Auswahl von Nutzer oder logischen Laufwerken erfolgt mit USER=n (n=0...15) und DRIVE=d (d=A...E). Für n oder d kann auch „ALL" zur Anzeige der Dateien aller Nutzer oder aller bis dahin aktiven logischen Laufwerke genutzt werden.
Andere Optionen dienen der Wiederholung des Tabellenkopfes nach einer gewissen Anzahl von Zeilen, der Anzeige aller Möglichkeiten von SDIR (HELP) u.a.m.
Auf die mögliche Ausgabe der Anzeigen auf den Drucker durch die Steuertaste CTRL-P sei hier nochmals verwiesen.
Beispiele:

– Sollen von allen Disketten und Nutzern alle COM-Files angezeigt werden, so sind folgende Angaben einzugeben:
sdir[drive=all,user=all] *.com
– Für die Anzeige aller R/O- und SYS-Dateien des Laufwerks C im kurzen Format ist folgende Eingabe erforderlich:
sdir[short ro sys] c:

PIP

Mit diesem transienten Kommando wird der Datenaustausch zwischen den peripheren Geräten des Bürocomputers realisiert (PIP=Peripheral Interchange Programm).

Mit Hilfe verschiedener Parameter lassen sich zusätzliche Funktionen beim Kopieren auslösen, zum Beispiel das Verketten von Dateien, das Kopieren von Teilen einer Datei oder die besondere Gestaltung von Drucklisten.

Formate

Es sind zwei grundsätzliche Formate möglich:
– PIP „kommandozeile"
– PIP.

Bei Verwendung des ersten Formates wird das Kommando PIP geladen, entsprechend der Kommandozeile abgearbeitet und beendet. Dabei ist zu beachten, daß alle in der Kommandozeile eingegebenen Kleinbuchstaben durch CCP (der Aufruf erfolgt ja im CCP-Status) in Großbuchstaben gewandelt werden. Dies ist besonders bei Zeichenketteneingaben bei Kopien von Dateiausschnitten zu beachten. Für diesen Fall sollte das zweite Format genutzt werden.

Beim zweiten Format wird nur PIP geladen. Mit dem Bereitschaftszeichen * wird die Arbeitsbereitschaft von PIP angezeigt, und der Bediener kann die Kommandozeile (nun nicht mehr im CCP-Status) eingeben. Jetzt könnte auch die Diskette mit dem Kommando PIP durch eine andere Diskette getauscht werden, die neue ist jedoch nur als Quelldiskette verwendbar! Abhilfe würde hier das Einbinden des PIP-Kommandos als neues residentes Kommando mit EXT schaffen. Beendet wird PIP im zweiten Format mit CTRL-C (Warmstart) oder einer Starttaste.

Kommandozeile

Sie hat das Format:
d:[zieldatei]=d:[quelldatei1],...[d:quelldatei n][p]

d bezeichnet hier nicht nur die logischen Laufwerke, sondern auch ein anderes logisches Gerät.

zieldatei, quelldatei entsprechen üblichen Dateibezeichnungen.

p steht für Parameter.

Logische Geräte

Bei PIP können neben den logischen Laufwerken auch folgende Gerätenamen für Tastatur, Bildschirm und Drukker angegeben werden:

CON: für Tastatur als Eingabegerät (Quelle) für Bildschirm als Ausgabegerät (Ziel)

LST: für Drucker als Ausgabegerät

PRN: für Drucker als Ausgabegerät, wobei im Unterschied zu LST: zusätzlich die Zeilen numeriert, Seitenvorschübe nach 60 Zeilen eingefügt und Tabulatoren nach je 8 Spalten gesetzt werden.

So bewirkt zum Beispiel das Kommando
pip lst:=con:
die Druckausgabe aller eingetasteten Zeichen (Schreibmaschinenfunktion).

Parameter

In der Kommandozeile können ein oder mehrere Parameter spezifiert werden, die in eckige Klammern einzuschließen und durch Leerzeichen voneinander zu trennen sind. Links des Gleichheitszeichens dürfen keine Parameter stehen.

Einige wichtige Parameter und ihre Bedeutung seien hier aufgeführt:

Gn
Es wird eine Datei des Nutzers n zum aktuellen Nutzer kopiert.

L
Großbuchstaben werden in Kleinbuchstaben umgesetzt.

U
Kleinbuchstaben werden in Großbuchstaben umgesetzt

R
Es können SYS-Dateien kopiert werden.

W
Es werden R/O-Dateien überschrieben, ohne daß vorher der Bediener darauf hingewiesen und um Erlaubnis gefragt wird.

SsCTRL-Z
Das Kopieren einer Datei beginnt erst, wenn die Zeichenkette s erkannt wurde.

QsCTRL-Z
Das Kopieren wird beim Erkennen der Zeichenkette s beendet.

Allgemeine Hinweise zum Gebrauch von PIP

– Beim Kopieren von Dateien werden Dateien gleicher Bezeichnung auf der Zieldiskette überschrieben. Ist die überschriebene Datei eine R/O-Datei, so wird der Bediener darauf hingewiesen, und er kann das Kopieren abbrechen. Die neue Datei auf der Zieldiskette erhält das R/W- und das DIR-Attribut. Nur wenn eine Datei mit SYS-Attribut überschrieben wird, erhält auch die neue Datei das SYS-Attribut.

– Die Quelldateien werden nie verändert!

– Während die Zieldatei nur eine eindeutige Dateibezeichnung oder nur die Laufwerksangabe haben kann, ist bei der Quelldatei eine mehrdeutige Dateibezeichnung zugelassen und auch sehr zweckmäßig. Beim Kopieren der dieser Bezeichnung entsprechenden Dateien werden diese einzeln bei ihrem Kopiervorgang aufgelistet.

– Sind mehrere Quelldateien angegeben, so werden sie in der Reihenfolge ihrer Angabe zu einer Zieldatei verkettet. Hierbei dürfen die Quelldateien nur eindeutige Dateibezeichnungen haben. Die Parameter sind für jede Quelldatei einzeln direkt hinter ihr anzugeben, nur der Parameter W ist einmal am Ende erforderlich.

– Beim Kopieren von Dateiausschnitten sind auch nur eindeutige Dateibezeichnungen zugelassen. Die mit dem Parameter Q oder/und S angegebenen Zeichenketten werden mitkopiert. Es löst jeweils die erste in der Datei auftretende Zeichenkette das Starten bzw. das Beenden des Kopiervorgangs aus.

– Während des Kopierens wird auf der Zieldiskette die Zieldatei mit dem Typ □□□ erzeugt. Erst wenn das Kopieren der Datei fehlerfrei abgeschlossen ist, wird der endgültige Typ im Dateiverzeichnis eingetragen.

Beispiele

– pip a:=b:*.*
Es werden alle Dateien des aktuellen Nutzers des logischen Laufwerks B auf das LW A kopiert (außer SYS-Dateien).

– pip b:test.neu=test.alt [sanf1CTRL-Z qend:CTRL-Z]
Vom aktuellen logischen Laufwerk wird aus der Datei TEST.ALT der Abschnitt von „ANF1" bis „End:" als neue Datei TEST.NEU auf der Diskette im Laufwerk B abgelegt.

– pip a:beisp=b:dat1,c:dat2
Die Datei DAT1 vom LW B und die Datei DAT2 vom LW C werden zur neuen Datei BEISP auf dem LW A verkettet.

SUBM
Mit diesem Kommando läßt sich eine Folge von Kommandos und Programmen (COM-Dateien) abarbeiten. Es sind alle Kommandos und Programme zulässig, die automatisch ausgeführt werden; das heißt, sie werden mittels Eingabe einer Kommandozeile gestartet und vollständig ausgeführt.
Diese Folge ist zunächst als eine Datei des Typs SUB zu erfassen; zum Beispiel mit dem Textprogramm TP 1520 (SCPX). Die einzelnen Kommandos oder Programme sind zeilenweise und in der gewünschten Abarbeitungsfolge in dieser Datei abzuspeichern. Leerzeilen sind verboten.
Zur Realisierung verschiedener Parameter, wie Dateiname, Dateityp, logisches Laufwerk, können diese in der Kommandodatei als Variable angegeben werden. Es sind maximal neun Variable möglich, die durch das Währungszeichen □(Code:24H) und die Ziffern 1...9 angegeben werden müssen. Diese Variablen werden dann beim Aufruf von SUBM durch aktuelle Werte ersetzt, die der Bediener als Parameter eingeben kann. Die Variable □1 wird durch den ersten Parameter ersetzt, die Variable □2 durch den zweiten Parameter usw.
Somit ist das Format von SUBM folgendes:
SUBM [d:]dateiname[parameterliste].
Die Parameter sind durch Leerzeichen getrennt einzugeben. Das aktuelle logische Laufwerk muß das LW A sein, während das Kommando SUBM und die SUB-Datei von einer anderen Diskette geladen werden können.
In der Kommandodatei kann als letztes Kommando auch das SUBM selbst stehen, so daß komplexere Folgen realisiert werden können. Nach Aufruf des SUBM wird zunächst eine temporäre Datei □□□.SUB auf der Diskette im Laufwerk A mit den substituierten Parametern gebildet. Es ist also noch freier Speicherplatz auf der Diskette im Laufwerk A erforderlich. Daran schließt sich die Abarbeitung der einzelnen Kommandos oder Programme an, als ob sie einzeln vom Bediener aufgerufen würden. Dementsprechend sind auch die Bildschirmanzeigen.
Mit CTRL-S kann der automatische Ablauf angehalten und wieder fortgesetzt werden.
Mit einer Start- oder Zeichentaste wird das aktuelle Kommando oder Programm abgebrochen und das nächste aufgerufen.
Beispiel:
Von drei logischen Laufwerken sollen auf einem alle Dateien eines Typs, auf einem zweiten alle Dateien eines anderen Typs gesammelt werden. Auf den jeweiligen anderen beiden Disketten sollen diese gesammelten Dateien gelöscht werden. Die Dateiverzeichnisse der drei logischen Laufwerke sind vorher und nachher auf dem Bildschirm anzuzeigen. Zu diesem Zweck ist zunächst die Kommandodatei unter der Dateibezeichnung SAMMEL.SUB zu erfassen:

```
sdir [drive=(□1,□2,□3)]
pip □2:=□1:*.□4
era □1:*.□4
pip □3:=□1:*.□5
era □1:*.□5
pip □2:=□3:*.□4
era □3:*.□4
pip □3:=□2:*.□5
era □2:*.□5
sdir [drive=(□1,□2,□3)]
```

Nun kann die Kommandodatei mit beliebigen Parametern aufgerufen werden:
subm sammel a c d txt erf
Für □1 wird nun a, für □2 c usw. ersetzt, und es werden folgende Kommandos ausgeführt:
SDIR[DRIVE=(A,C,D)]
Anzeige der Dateiverzeichnisse der LW A, C, D
PIP C: = A:* .TXT
Alle Dateien vom Typ TXT des LW A werden auf das LW C kopiert.
ERA A:*.TXT
Alle Dateien vom Typ TXT werden auf dem LW A gelöscht.
PIP D:=A:*.ERF
Auf das LW D kommen alle Dateien des Typs ERF vom LW A.
usw.

Am Ende sind alle .TXT-Dateien nur noch auf LW C, alle .ERF-Dateien auf LW D.

XSUB
Um bei der Abarbeitung von Kommandodateien auch Tastatureingaben einfügen zu können, ist am Anfang der SUB-Datei das Kommando XSUB zu setzen.
Wird solch eine Kommandodatei abgearbeitet, so wird XSUB direkt vor CCP gelagert und übernimmt die Steuerung.
In der Kommandodatei dürfen nun als selbständige Zeilen Eingabeinformationen zum davorstehenden Kommando oder Programm enthalten sein, die sonst mittels der BDOS-Funktion 10 (s. 5.6.1.) während der einzelnen Abarbeitung dieser Kommandos oder Programme vom Bediener eingegeben werden. Diese Eingabewerte können auch Variable sein, die beim Aufruf der Kommandodatei durch die Parameter ersetzt werden.
Die Forderung, daß alle Tastatureingaben mittels BDOS-Funktion 10 realisiert sind, wird durch alle transienten Kommandos erfüllt

5.5.4.
Allgemeine Fehlermeldungen

Neben den kommandospezifischen Fehlern (z. B. falsche oder unzulässige Dateibezeichnungen, falsche Parameterangaben) werden von SCPX folgende allgemeingültige Fehlermeldungen erzeugt.
Fehler bei Diskettenarbeit
Es werden vier verschiedene Fehler, die bei der Diskettenarbeit auftreten können, von SCPX gemeldet (d steht stets für das logische Laufwerk, bei dem der Fehler auftritt):
SCPX ERR ON d: BAD SECTOR
Diese Meldung kann entstehen
– durch Lese- und Schreibfehler
– bei anderem Diskettenformat als es SCPX auf diesem Laufwerk erfordert
– durch fehlende Kompatibilität zwischen dem Laufwerk, auf dem die zu lesende Diskette erstellt wurde, und dem mit d angezeigten Laufwerk
– bei unsachgemäßer Behandlung der Diskette.
Die Fortsetzung nach dieser Meldung kann mit CTRL-C oder einer beliebigen Taste erfolgen. Mit CTRL-C wird das Kommando abgebrochen und ein Warmstart ausgeführt; die fehlerhafte

Diskette kann nun ausgetauscht werden. Mit einer anderen Taste wird das Kommando fortgesetzt, was aber bei Schreibfehlern zu weiteren Zerstörungen von Disketteninformationen führen kann!

SCPX ERR ON d: FILE R/O

Wird versucht, eine Datei mit dem R/O-Attribut zu überschreiben oder zu löschen, so wird diese Fehlermeldung auf dem Bildschirm angezeigt. Eine Weiterarbeit ist nicht möglich; mit einer beliebigen Taste wird das Kommando abgebrochen und ein Warmstart ausgelöst.

SCPX ERR ON d: R/O

Diese Fehlermeldung erscheint auf dem Bildschirm, wenn auf eine Diskette schreibend zugegriffen wird, diese jedoch den R/O-Status hat. Der R/O-Status kann durch STAT erzeugt worden sein, oder es wurde nach einem Diskettenwechsel kein Warmstart ausgeführt.
Mit einer beliebigen Taste hat der Bediener diese Meldung zu quittieren. Danach wird ein Warmstart vom System ausgelöst, womit unter anderem auch der R/W-Status für die betreffende Diskette hergestellt wird.

SCPX ERR ON d: SELECT

Wenn auf ein logisches Laufwerk zugegriffen werden soll, das im betreffenden System nicht existiert, bringt SCPX eine Fehlermeldung. Dies wird zum Beispiel stets bei der Anwahl der logischen Laufwerke F, G usw. passieren, da diese Laufwerke bei SCPX nicht möglich sind.
Mit einer beliebigen Taste geht das System über einen Warmstart in den Systemgrundstand.

Druckerfehler

Tritt beim Drucker ein Fehler auf, so wird das durch eine LED auf der Tastatur signalisiert. Es wird entweder vom Bediener das Kommando mit CTRL-C abgebrochen, oder nach Fehlerbehebung am Drucker kann die Abarbeitung des Kommandos mit einer beliebigen Taste fortgesetzt werden.
Bei den Druckern mit internen Puffern leuchtet die LED auch dann, wenn der Drucker keine Zeichen mehr in seinen Puffer übernehmen kann. Da aber der Drucker dabei Zeichen druckt, ist das vom Bediener nicht als Druckerfehler, sondern nur als Wartezustand zu deuten.
Oft entstehen Fehler bei der Arbeit mit dem Drucker, wenn kritiklos alle Steuerzeichen, die für den Bildschirm gedacht sind, auch zum Drucker ausgegeben werden (z. B. mit CTRL-P).

5.6.
E/A-System

5.6.1.
Vorbemerkungen

Das Erarbeiten von Programmen, die unter Steuerung von SCPX laufen sollen, ist dann erforderlich, wenn keine entsprechende Standardsoftware angeboten wird. Die zur Verfügung stehende Programmentwicklungssoftware (s. 5.7.) bietet dazu die Möglichkeit des komfortablen Programmierens im Assemblersprachniveau (U 880 -Assemblersprache). Damit können bezüglich Laufzeit und Speicherplatzbedarf die effektivsten Programme für SCP geschrieben werden. Das Einarbeiten in die Assemblersprache erfordert jedoch einen hohen Lernaufwand, außerdem sind einige Kenntnisse über die Hardware nötig.
Aus diesem Grund sollte verstärkt auf die Programmierung in höheren Programmiersprachen orientiert werden (z. B. PASCAL, FORTRAN, C, BASIC). Hierfür werden im SCP entsprechende Compiler und Interpreter angeboten. Die so erstellten Programme sind im allgemeinen einfacher und schneller zu erarbeiten, zu testen, zu betreuen und zu dokumentieren. Die Programmlaufzeit und der Speicherbedarf werden jedoch zum Teil wesentlich größer sein, was aber in vielen Fällen von untergeordneter Bedeutung ist. Ein weiterer Vorteil höherer Sprachen liegt darin, daß Programme dieser Sprachen kompatibel für andere Rechner gestaltet werden können, wenn diese Rechner über Interpreter oder Compiler mit gleichem Sprachniveau verfügen.
In diesem Abschnitt wird das für die Programmierung mittels Assemblersprache unbedingt erforderliche E/A-System vorgestellt. Es ermöglicht den einfachen und standardisierten Zugriff auf die Geräte, Tastatur, Bildschirm, Drukker und Diskettenlaufwerke. Unter bestimmten Voraussetzungen können auch der Lochbandleser und der Lochbandstanzer (K 6200) über das E/A-System von SCPX aufgerufen werden.
Für das normale Programmieren sind — neben der Beherrschung der CPU-Sprache des U 880 — Kenntnisse des logischen Teils des E/A-Systems (BDOS) erforderlich. Zur Lösung spezieller Aufgaben eines Systemprogrammierers wird auch die physische Ebene von SCPX (BIOS) erläutert. Auf die Parameterübergabe bei einem Programmaufruf wird im letzten Teil dieses Abschnittes eingegangen.

5.6.2.
Logisches E/A-System (BDOS)

Das logische E/A-System realisiert den Zugriff auf die logischen Geräte Konsole (Tastatur und Bildschirm), Drucker und Disketten-Laufwerke. Für BDOS ist es unwichtig, welche tatsächlichen physischen Geräte sich hinter diesen verbergen.
Bei Anwendung des Dienstprogramms EML.COM werden auch ein Lochbandleser und ein Lochbandstanzer in das E/A-System integriert.
Die Gesamtheit von BDOS wird in die Funktionen 0 bis 36 unterteilt. Zum Aufruf der einzelnen Funktionen gelten folgende Festlegungen:

- Die Funktionsnummer ist im Register C einzutragen.
- Der Aufruf der Funktion erfolgt durch einen Unterprogrammaufruf mit der Adresse 5.

Dort wurde beim Kaltstart ein Sprung an BDOS eingetragen. Von einem Programm sollte niemals BDOS direkt (auf D000H) aufgerufen werden.

- Für zahlreiche Funktionen sind weitere Zusatzinformationen erforderlich. Sie sind im Registerpaar DE oder nur in E zu übergeben.

Damit ergibt sich folgendes Schema:

LD C,n n ... Funktionsnummer
[LD DE,xxxx]
xxxx ... Zusatzinformationen
CALL 5

Mittels BDOS können alle Register außer IX und IY verändert werden, so

daß entsprechende Maßnahmen (z. B. PUSH-Befehle vor und POP-Befehle nach dem BDOS-Aufruf) im Programm erforderlich sind.
BDOS legt einen eigenen Stack an.
Zahlreiche Funktionen senden eine Rückmeldung über die Ausführung der Funktion. Diese wird im Register A und/oder in Registerpaar HL übergeben. Abweichungen davon werden bei den einzelnen BDOS-Funktionen erläutert.

BDOS-Funktion 0 : System-Neustart

Diese Funktion hat eine spezielle Wirkung. Sie beendet das Programm durch das Auslösen eines Warmstarts. Die identische Wirkung wird mittels Sprung an die Adresse 0 erreicht, da dort beim Kaltstart ein Sprung zur Warmstartroutine (in BIOS) eingetragen wurde.
Nach dieser Funktion oder einem Sprung an 0 befindet sich das System wieder im Grundzustand, den die Anzeige des aktuellen logischen Laufwerkes und der Bereitschaftszeichen kennzeichnen.

BDOS-Funktion 1 : Konsole-Eingabe

Es wird das nächste Zeichen von der Tastatur in das Register A übernommen. Das eingegebene 7-Bit-Zeichen wird auf dem Bildschirm an der aktuellen Kursorposition ausgegeben (Echo). Liegt kein Zeichen an, so wird gewartet. Zulässige Steuerzeichen sind CR (Wagenrücklauf), LF (Zeilenschaltung), HT (8–er Tabulation) und BS (Backspace). Andere Steuerzeichen werden ignoriert!

BDOS-Funktion 2 : Konsole-Ausgabe

Das im Register E befindliche Zeichen wird auf den Bildschirm an die aktuelle Kursorposition ausgegeben.
Neben den darstellbaren Zeichen sind unter anderem folgende wichtige Steuerzeichen (Code hexadezimal) für die Bildschirmsteuerung realisiert:

01: Kursor an Bildschirmanfangsposition
08: Kursor ein Zeichen nach links (BS)
0A: Zeilenschaltung (LF)
0C: Löschen des Bildschirmes und Kursor an den Anfang des Bildschirms
0D: Kursor an Zeilenanfang (CR)
14: Löschen des Bildschirms ab aktueller Kursorposition
15: Kursor ein Zeichen nach rechts
16: Löschen ab Kursorposition bis Zeilenende
18: Löschen der ganzen Zeile und Kursor an Zeilenanfang
1A: Kursor eine Zeile nach oben
1B: Einleiten einer beliebigen Kursorpositionierung.
Die folgenden zwei Bytes geben die Zeile (0..15, bzw. 0..23) und die Spalte (0..63, bzw. 0..79) an. Das werthöchste Bit ist bei beiden Bytes zusätzlich zu setzen.
82: Kursor ein
83: Kursor aus.

Sobald die letzte Bildschirmposition überschritten wird, zum Beispiel durch eine Zeilenschaltung, erfolgt ein Rollen des Bildschirms um eine Zeile nach oben.

BDOS-Funktion 3 : Lochbandeingabe

Diese Funktion und die Funktion 4 sind nur realisiert, wenn vorher das Hilfsprogramm EML(.COM) aufgerufen wurde. Damit erfolgte eine Erweiterung des E/A-Systems von SCPX um diese Funktionen. Sonst sind die Funktionen wirkungslos.
Mit der Funktion 3 wird vom Lochband ein Zeichen in das Register A gelesen (7-Bit-Code, ohne Paritätsbit).

BDOS-Funktion 4 : Lochbandausgabe

Das im Register E vorhandene 7-Bit-Zeichen wird zum Lochbandstanzer ausgegeben. Es erfolgt keine Paritätsbitbildung!

BDOS-Funktion 5 : Druckerausgabe

Das Zeichen, das im Register E steht, wird an den Drucker ausgegeben. Es können neben den druckbaren Zeichen auch alle Steuerzeichen oder Steuerzeichenfolgen übertragen werden, die der Drucker verarbeiten kann.

BDOS-Funktion 6 : Direkte Konsolen-Ein-/Ausgabe

Wenn das Register E den Wert FFH hat, so erfolgt die Übernahme eines Zeichens von der Tastatur in das Register A. Wurde keine Taste unmittelbar vor der Funktion betätigt, so wird nicht gewartet, und im Register A wird der Wert 0 eingetragen. Ein weiterer Unterschied zur Funktion 1 besteht darin, daß die eingegebenen Zeichen nicht auf dem Bildschirm „geechot" werden. Somit können auch Steuertasten eingegeben und im Programm selbst bewertet werden.

Tab. 3 Charakterisierung der Bytes des File-Control-Blocks

Byte	Kurzbezeichnung	Bedeutung
0	dr	Code des logischen Laufwerkes, wo sich die Datei befindet oder aufgezeichnet werden soll 0 ... aktuelles Laufwerk (s.Fkt. 14) 1 ... LW A 2 ... LW B 3 ... LW C 4 ... LW D 5 ... LW E
1 ... 8	f1 ... f8	Dateiname
9 ... 11	t1 ... t3	Dateityp Die Bits 7 von t1 und t2 legen das Dateiattribut fest. Sind diese Bits z. B. gesetzt, so handelt es sich um eine R/O-(t1) und SYS-Datei (t2).
12	ex	aktuelle Extentnummer
13,14		reserviert für SCPX
15	rc	Anzahl Records des aktuellen Extent (0 ... 80H)
16 ... 31		reserviert für SCPX
32	cr	aktueller Record für den nächsten sequentiellen Zugriff
33 ... 35	r0, r1, r2	nur für Direktzugriff erforderlich r0 ... L-Teil r1 ... H-Teil der Recordadresse r2 ... für Überlauf

Alle anderen Werte im Register E führen zur Bildschirmausgabe dieses Wertes an der aktuellen Kursorposition.

BDOS-Funktionen 7 und 8

Diese beiden Funktionen werden unter SCPX nicht genutzt und werden nur aus Kompatibilitätsgründen mitgeführt.

BDOS-Funktion 9:
Zeichenkettenausgabe

Hiermit werden Zeichenketten auf den Bildschirm ausgegeben. Die Adresse dieser Kette ist im Registerpaar DE vorzugeben, das Ende der Zeichenkette wird durch das Zeichen ¤ (Code:24H) festgelegt. In der Zeichenkette können auch Steuerzeichen enthalten sein. Da durch ein vorhergehendes CTRL-P auch eine gleichzeitige Druckausgabe erfolgen kann, sollten nur die Steuerzeichen verwendet werden, die jeder Drukker akzeptiert; zum Beispiel CR, LF, BS.

BDOS-Funktion 10:
Zeichenketteneingabe

Mit dieser Funktion kann eine Zeichenkette von der Tastatur in den Hauptspeicher eingegeben werden. Im Registerpaar DE ist dafür die Adresse des Zeichenkettenpuffers vorzugeben.

Im ersten Byte dieses Puffers ist vorher die maximal eingebbare Zeichenzahl (1..255) einzutragen. Das zweite Byte enthält nach Abschluß der Eingabe die tatsächliche Anzahl eingegebener Zeichen.

Ab der dritten Position werden die einzelnen Zeichen in der Reihenfolge ihrer Eingabe abgelegt. Der Puffer wird nicht automatisch gelöscht, so daß nur die Zeichen entsprechend der Angabe im Byte 2 gültig sind.

Beendet wird die Zeicheneingabe durch
- Erreichen der Maximalanzahl
- eine Starttaste
- CTRL-J
- CTRL-M.

Der Kursor wird danach auf die erste Position der Zeile gesetzt. Eine große Zahl von Steuerzeichen ermöglicht ein einfaches zeilenorientiertes Editieren. Diese Steuerzeichen sind bereits unter 5.5.1. erläutert, da im CCP die Kommandoeingabe über diese BDOS-Funktion realisiert wird.

Wird innerhalb dieser Funktion die Steuerfunktion CTRL-P ausgelöst, so bleibt die Wirkung des gleichzeitigen Druckes bei Bildschirmausgaben auch für die folgenden BDOS-Funktionen 2, 9 und 10 erhalten, bis ein zweites CTRL-P oder ein Warmstart dies wieder aufhebt.

BDOS-Funktion 11:
Abfrage Konsolenstatus

Mit dieser Funktion kann geprüft werden, ob ein Zeichen von der Tastatur anliegt. Wird im Register A eine 0 zurückgemeldet, so ist kein Zeichen von der Tastatur gemeldet. Im anderen Falle enthält A den Wert 1. Eine Übernahme des Zeichens kann dann mit den BDOS-Funktionen 1 oder 10 erfolgen.

BDOS-Funktion 12:
Abfrage Versionsnummer

Mittels dieser Funktion werden im Register H der Wert 0, im Register L der Wert 22H eingetragen. Dies erfolgt – wie bei Funktion 7 und 8 – aus Kompatibilitätsgründen.

BDOS-Funktion 13:
Rücksetzen des Diskettensystems

Diese Funktion ist erforderlich, um in einem Programm Disketten wechseln zu können. Ein Warmstart, der einen Diskettenwechsel ebenfalls absichert, führt ja zum Programmabbruch.

Das logische Laufwerk A wird zum aktuellen Laufwerk, alle anderen Laufwerke erhalten den R/W-Status (s. Funktion 28), und die aktuelle Adresse für den Recordpuffer im Hauptspeicher wird auf 80H eingestellt (s. Funktion 26).

File Control-Block

Für viele der nun folgenden Funktionen, die der Realisierung der Diskettenarbeit im E/A-System dienen, wird ein Parameterblock benötigt, der alle dafür wichtigen Informationen enthält. Dieser Parameterblock wird File-Control-Block (FCB) genannt, seine Adresse im Hauptspeicher muß im Registerpaar DE vorgegeben werden. Der FCB besteht aus 33 Byte bei sequentiellem Zugriff und aus 36 Byte bei Direktzugriff auf die betreffende Datei. Die Bedeutung der Bytes ergibt sich aus Tab. 3. Vor Dateioperationen ist stets ein derart aufgebauter FCB erforderlich. Vom Programmierer sind die Bytes 0–11 entsprechend obiger Angaben einzutragen, die anderen Bytes sind im allgemeinen auf 0 zu setzen.

BDOS-Funktion 14:
Aktuelles logisches Laufwerk auswählen

Hiermit läßt sich eines der logischen Laufwerke als aktuelles Laufwerk einstellen. Der Wert im Register E gibt an, welches ausgewählt wird:

E = 0 ... Laufwerk A
E = 1 ... Laufwerk B usw.

Das ausgewählte Laufwerk bleibt bis zur nächsten Funktion 14 oder bis zum nächsten Warmstart das aktuelle logische Laufwerk. Es wird stets angesprochen, wenn im FCB, Byte 0, der Wert 0 steht.

BDOS-Funktion 15:
Datei eröffnen (Open)

Im Registerpaar DE ist die Adresse des FCB vorzugeben. Es wird die damit benannte Datei (Bytes f1 ... t3), die auf dem durch das Byte 0 des FCB ausgewählten logischen Laufwerk existieren muß, aktiviert. Einige Werte aus der Dateieintragung werden in den FCB übernommen, so daß anschließend Records der Datei gelesen oder geschrieben werden können.

Ist anschließend ein sequentieller Zugriff auf den ersten Record der Datei vorgesehen, so ist das Byte cr im FCB auf 0 zu setzen! Steht nach dieser Funktion im Register A der Wert FFH, so wurde eine Datei mit der vorgegebenen Bezeichnung nicht gefunden. Die Dateieröffnung war positiv, wenn in A einer der Werte 0, 1, 2 oder 3 steht.

BDOS-Funktion 16:
Datei schließen (Close)

Auch hier ist in DE die Adresse des FCB vorzugeben. Die in ihm gespeicherten Informationen werden durch diese Funktion in das Dateiverzeichnis übernommen. Es kann nur eine Datei geschlossen werden, die vorher eröffnet worden war (Funktion 15 oder 22). Wurde eine Datei nicht verändert, so ist ein Schließen nicht erforderlich. In allen anderen Fällen würde ein fehlendes Close zum Datenverlust führen.

Die Rückmeldungen sind die gleichen wie beim Eröffnen einer Datei.

BDOS-Funktion 17:
Suchen nach erster Eintragung

Mit dieser Funktion werden die Eintra-

gungen im Dateiverzeichnis zugänglich gemacht.
Das Registerpaar DE enthält wieder die FCB-Adresse.
Das Dateiverzeichnis wird von Anfang an nach der ersten Eintragung einer Datei durchsucht, die den FCB-Eintragungen dr ... t3 entspricht. Es sind auch Fragezeichen in den Positionen f1 ... t3 möglich, diese Positionen werden beim Vergleich nicht herangezogen.
Ein Fragezeichen auf der ersten Stelle des FCB bewirkt, daß nach dem ersten Record des Dateiverzeichnisses des aktuellen Laufwerks gesucht wird.
Steht nach dieser Funktion in A der Wert 0, 1, 2 oder 3, so wurde die Eintragung gefunden, sonst enthält A ein FFH.
Die ermittelte Eintragung (32 Byte lang) wird in einem durch die Funktion 26 vorgegebenen Puffer im Hauptspeicher abgelegt. Wurde die Funktion 26 vorher nicht ausgeführt, beginnt der Puffer auf der Adresse 80H. Der Puffer ist zur Aufnahme eines Records 128 Byte groß. Je nach dem Wert von A ist die Eintragung ab dem ersten Byte (A=0), ab dem 33. Byte (A=1), ab dem 65. Byte (A=2) oder ab dem 97. Byte (A=3) innerhalb des Puffers zu finden.

BDOS-Funktion 18:
Suchen nach nächster Eintragung
Diese Funktion ist im Anschluß an die Funktion 17 zu programmieren, wenn nach weiteren Eintragungen ab der aktuellen Position im Verzeichnis gesucht werden soll. Dies ist dann sinnvoll, wenn im FCB, dessen Adresse auch hier im Registerpaar DE vorzugeben ist, Fragezeichen enthalten sind oder die Datei mehrere Eintragungen belegt. Nach der Funktion 17 ist die Funktion 18 so oft aufzurufen, bis im Register A der Wert FFH zurückgemeldet wird.
Ein Fragezeichen auf der Position 0 bewirkt, daß alle Eintragungen der Diskette nacheinander als gefunden erklärt werden.
Für die Rückmeldungen dieser Funktion gelten die gleichen Bedingungen wie bei der Funktion 17.
Die Funktion 18 darf nur nach der Funktion 17 aufgerufen werden!

BDOS-Funktion 19:
Löschen einer Datei
Mit dieser Funktion können eine oder mehrere Dateien gelöscht werden.
Im Registerpaar DE ist wieder die FCB-Adresse zu übergeben. Im Dateinamen und im Dateityp sind Fragezeichen zugelassen, so daß mehrere Dateien mit einem Aufruf gelöscht werden können.
Ein erfolgreiches Löschen, das heißt die betreffende Datei oder Dateien wurden gefunden, wird mit den Werten 0, 1, 2 oder 3 im Register A gemeldet.
Im anderen Falle steht im Register A ein FFH.

BDOS-Funktion 20:
Sequentielles Lesen
In DE ist die FCB-Adresse anzugeben. Es wird mit dieser Funktion ein Record der betreffenden Datei in den Speicher eingelesen. Die Anfangsadresse dafür ist entweder 80H, oder sie wurde mit der BDOS-Funktion 26 vorgegeben. Im Byte cr des FCB steht die Adresse des Records in der Datei. Soll der erste Record der Datei gelesen werden, so muß cr den Wert 0 enthalten. Der Programmierer hat dies beim Aufbau des FCB zu berücksichtigen. Nach dem Lesen wird cr um eins automatisch erhöht. Ist eine Extentgrenze erreicht, wird von BDOS automatisch auf den nächsten Extent geschaltet und cr auf Null gestellt.
Wird im Register A eine Null zurückgemeldet, so war das Lesen erfolgreich, bei jedem anderen Wert wurde das Dateiende erreicht.

BDOS-Funktion 21:
Sequentielles Schreiben
Von der Adresse 80H oder der mit der Funktion 26 vorgegebenen Adresse beginnend wird ein Record in die Datei geschrieben. Diese ist durch ein FCB vorgegeben, dessen Adresse in DE vorher einzutragen ist. Das Byte cr gibt wieder die Position innerhalb der Datei an, es wird nach dem Schreiben des Records ebenfalls um eins erhöht. Das Umschalten auf den nächsten Extent erfolgt wie auch schon beim Lesen automatisch.
Ein erfolgreiches Schreiben wird mit 0 im Register A gemeldet. Ist die Diskette oder das Dateiverzeichnis (beim Überschreiten einer Eintragungsgrenze) voll, so wird ein Wert ungleich 0 in A von BDOS eingetragen.
Auf keinen Fall darf nach dem Schreiben einer Datei die Close-Funktion fehlen, da es sonst zu Dateiverlusten kommen kann.
Lese- und Schreibfehler führen zu der Bildschirmausschrift
SCPX ERR ON d: BAD SECTOR
Mit d wird wieder das logische Laufwerk angegeben.
Soll trotzdem im Programm weitergearbeitet werden, so ist eine Starttaste zu betätigen. Der Fehler wird dann ignoriert. Dabei kann es aber bei Schreibfehlern zu weiteren Datenverlusten kommen. Mit CTRL-C kann hier der Bediener einen Warmstart auslösen und dadurch das Programm abbrechen.

BDOS-Funktion 22:
Anlegen einer Datei
Diese Funktion dient zum Anlegen einer neuen Datei. In DE ist die FCB-Adresse der Datei anzugeben.
War die Funktion erfolgreich, das heißt, die Datei wurde in das Dateiverzeichnis aufgenommen, dann steht im Register A ein Wert von 0 bis 3. Der Wert FFH sagt dagegen aus, daß im Verzeichnis kein Platz mehr dafür war.
Da beim Anlegen der Datei keine Prüfung auf eine bereits bestehende Datei gleicher Bezeichnung erfolgt, sollte vorher entweder ein Test mit der Funktion 15 (Open) oder die Funktion 19 (Löschen der Datei) ausgeführt werden.
Nach dem Anlegen einer Datei ist ein Open nicht mehr erforderlich.

BDOS-Funktion 23:
Umbenennen einer Datei
Für diese Funktion ist im Registerpaar DE ein etwas anders aufgebauter FCB zu adressieren. Die ersten 12 Bytes enthalten im üblichen Format die alte Dateibezeichnung, die Bytes 16 bis 26 den neuen Dateinamen und Dateityp. Das logische Laufwerk, das die umzubenennende Datei enthält, wird nur im Byte 0 angegeben. Die restlichen Bytes sind auf 0 zu setzen.
Nach erfolgreicher Ausführung enthält Register A den Wert 0, 1, 2 oder 3. Wurde die Datei nicht gefunden, so ist in A der Wert FFH eingetragen.

BDOS-Funktion 24:
Abfrage nach on-line-Laufwerken
Mit dieser Funktion läßt sich feststellen, mit welchem der logischen Laufwerke nach dem Kaltstart, Warmstart oder der Funktion 13 bereits gearbeitet wurde.
Dies wird in den Registern A und L mitgeteilt, wobei je eins der Bits 0–4 dabei den Laufwerken A–E zugeordnet ist.
Nach dem Kaltstart ist zum Beispiel nur das Bit 0 in A und L gesetzt.
BDOS-Funktion 25:
Abfrage nach aktuellem Laufwerk
Im Register A wird gemeldet, welches der logischen Laufwerke das aktuelle ist. Der Wert 0 steht für das LW A, der Wert 1 für das LW B usw.
BDOS-Funktion 26:
Einstellen der Adresse für den Recordpuffer
Der Datentransport von und zur Diskette erfolgt stets über einen Puffer im Hauptspeicher, der einen Record (128 Byte) aufnehmen kann. Die Adresse dieses Puffers ist nach dem Kaltstart, einem Warmstart und nach der Funktion 13 mit 80H festgelegt.
Mit dieser Funktion kann eine beliebige andere Adresse eingestellt werden. Sie ist im Registerpaar DE vorzugeben.
BDOS-Funktion 27:
Bereitstellen der Adresse des Belegungsvektors
Diese nur für den Systemprogrammierer interessante Funktion liefert im Registerpaar HL die Adresse des Belegungsvektors des aktuellen logischen Laufwerkes. Somit sind Informationen über die Auslastung der Diskette zu erhalten.
BDOS-Funktion 28:
Diskette mit Schreibschutz versehen
Nach dieser Funktion kann auf das aktuelle logische Laufwerk nicht mehr geschrieben werden. Ein Schreibversuch auf diese Diskette wird mit der Mitteilung SCPX ERR ON d:R/0
auf dem Bildschirm beantwortet.
Mit der Funktion 13 kann dieser R/O-Zustand der Diskette wieder aufgehoben werden.
BDOS-Funktion 29:
Abfrage nach Laufwerk mit Schreibschutz
In den Registern A und L wird bitweise vermerkt, welche der logischen Laufwerke im R/O-Zustand sind. Die Zuordnung von Bit und Laufwerk ist die gleiche wie bei der Funktion 24.
BDOS-Funktion 30:
Setzen Dateiattribut
Mit der Funktion werden die Dateiattribute R/W oder R/O und DIR oder SYS gesetzt. Die betreffende Datei ist durch einen FCB, dessen Adresse im Registerpaar DE einzutragen ist, vorzugeben. Ist das Bit 7 von t1 gesetzt, so wird das Dateiattribut R/O, sonst R/W. Mit einem gesetzten Bit 7 von t2 wird die Datei eine SYS-Datei, im anderen Falle eine DIR-Datei.
BDOS-Funktion 31:
Bereitstellen der Adresse der Laufwerksparameter
Diese Funktion ist nur für Systemprogrammierer nützlich. Mit ihr wird die Adresse eines Parameterblockes in HL übergeben, in dem Informationen über das aktuelle logische Laufwerk enthalten sind.
BDOS-Funktion 32:
Einstellen / Abfrage des Nutzercodes
Trägt man vor dem Aufruf dieser Funktion in das Register E den Wert FFH, so wird der aktuelle Nutzercode im Register A rückgemeldet.
Ist der Wert in E ein anderer, so werden die unteren fünf Bit dieses Wertes als neuer Nutzercode (0 ... 31) eingestellt.
Es ist zu beachten, daß mit dem residenten Kommando USER nur die Nutzer 0 bis 15 aufgerufen werden können! Darum sollte man sich im Normalfall auch bei dieser BDOS-Funktion auf diese Nutzer beschränken.
Direktzugriff auf Dateien
Für den Direktzugriff sind die Bytes r0, r1 und r2 des FCB erforderlich. Vor jedem Zugriff sind in r0 der L-Teil und in r1 der H-Teil der Adresse des Records in der Datei vorzugeben. Das Byte r2 ist auf 0 zu setzen. Mit r0=r1=0 wird der erste Record der Datei adressiert. Bezüglich des Datenpuffers gilt das gleiche wie beim sequentiellen Zugriff.
Im Gegensatz zum sequentiellen Zugriff erfolgt nach dem Lesen oder dem Schreiben des Records kein Erhöhen der Recordadresse. Das Byte cr (und ex) wird auch auf den mit r0 und r1 adressierten Record eingestellt, so daß sich ein anschließender sequentieller oder direkter Zugriff auf den selben Record beziehen würde. Ein Übergang vom direkten zum sequentiellen Zugriff ist unter Beachtung dieser Tatsache stets möglich. Bei den vorher erforderlichen BDOS-Funktionen 15 oder 22 ist darauf zu achten, daß im adressierten FCB das Byte ex auf 0 gesetzt ist.
BDOS-Funktion 33:
Lesen im Direktzugriff
In DE ist die FCB-Adresse vorzugeben.
Das Lesen war fehlerfrei, wenn im Register A eine 0 steht.
Da nicht alle Records innerhalb einer Direktzugriffsdatei existieren müssen, kann es zu verschiedenen Fehlern und dementsprechenden Meldungen im Register A kommen:
A = 1 Dieser Satz war nicht geschrieben worden.
A = 3 Ein vorhergehendes Schreiben konnte nicht fehlerfrei beendet werden.
A = 4 Der mit r0 und r1 adressierte Extent existiert nicht.
A = 6 Die eingestellte Recordadresse liegt außerhalb der Diskettenkapazität (r2 > 0).
Es ist zu beachten, daß, sobald ein Record innerhalb eines Blockes im Direktzugriff geschrieben wurde, alle anderen 15 Records in diesem Block als geschrieben zählen. War der betreffende Record der adreßhöchste der Datei, so gilt das nur bis zu diesem.
BDOS-Funktion 34:
Schreiben im Direktzugriff
Hier gilt sinngemäß das unter der Funktion 33 Gesagte. Falls durch die Recordadressenvorgabe ein neuer Extent oder eine neue Eintragung für die Datei erforderlich ist, erfolgt das automatisch.
Neben den Fehlermeldungen A=3 und A=6, die mit den Fehlern vom Lesen identisch sind, kann es auch zum Fehlercode A=5 kommen. Dieser signalisiert, daß keine Eintragung im Dateiverzeichnis mehr frei war. Wenn auf der Diskette kein freier Block mehr existiert, so wird in A der Wert 2 gemeldet.
BDOS-Funktion 35:
Bereitstellen der nächsten freien Recordposition
Mit dieser Funktion kann die nächste

freie Recordadresse einer Datei abgefragt und in den Bytes r0 und r1 eingetragen werden. Während für sequentiell erfaßte Dateien damit auch die Größe der Datei (Anzahl der Records) feststellbar ist, müssen bei Direktzugriffsdateien eventuelle Löcher in der Datei beachtet werden.

Nach dieser Funktion ist ein problemloses Erweitern einer Datei im Direktzugriff möglich.

In DE ist die FCB-Adresse vorzugeben.

BDOS-Funktion 36:
Bereitstellen aktuelle Recordposition

Mit dieser Funktion wird nach einem sequentiellen Lesen oder Schreiben die Adresse des zuletzt gelesenen oder geschriebenen Records in r0 und r1 eingetragen.

Damit ist zum Beispiel ein leichter Übergang vom sequentiellen zum direkten Zugriff möglich.

5.6.3. Physisches E/A-System (BIOS)

Der Teil BIOS von SCPX enthält die programmtechnische Realisierung der Ansteuerung der peripheren Geräte und ist in einzelne, einfache Rufe unterteilt. Am Anfang von BIOS steht eine Tabelle von Sprungbefehlen, die zu den verschiedenen BIOS-Rufen führen.

Die einzelnen Rufe werden im folgenden in der Reihenfolge der zugehörigen (3-Byte-)Sprungbefehle erläutert. Ausgehend von der Adresse des Sprungbefehls zur Warmstartroutine, die auf den Speicherzellen 1 und 2 abgelegt ist, können so alle Adressen der einzelnen Routinen ermittelt werden. Der Sprungbefehl zur Warmstartroutine ist der zweite in der Tabelle.

Bei den BIOS-Rufen für die Diskettenarbeit wird sich generell und unabhängig von der physischen Sektorgröße stets auf einen Sektor von 128 Byte bezogen!

Es sollte jedoch nur in Ausnahmefällen die Programmierung mit BIOS-Rufen angewendet werden.

Kaltstartroutine

Nach dem Systemladen mittels SYL wird diese Routine im BIOS aufgerufen. Hier werden nun Grundeinstellungen für die CPU (I-Register, Stackpointer) und den Hauptspeicher realisiert, Kennungen für den Betriebsbeginn gesetzt und der Tastaturanschluß initialisiert.

Nach der Systemanzeige ("SCPX 1526 ...") werden Sprungbefehle auf die Adresse 0 zur Warmstartroutine und auf die Adresse 5 zu BDOS eingetragen. Die Adresse des Recordpuffers für Dateiarbeit wird auf 80H eingestellt.

Die Routine, deren Aufruf von einem Programm nicht sinnvoll ist, endet mit dem Aufruf von CCP.

Warmstartroutine

Diese Routine wird von Programmen oder Kommandos mittels CTRL-C (während der BDOS-Funktion 10 eingegeben), durch einen Sprung an die Adresse 0 oder durch die BDOS-Funktion 0 aufgerufen.

In dieser Routine wird zunächst CCP von der Systemdiskette im LW A nachgeladen. Auf fehlerfreies Einlesen wird geprüft. Fehler werden mit der Meldung "NO SYSTEM" auf dem Bildschirm ausgewiesen und lassen sich nur durch einen Kaltstart beheben.

Das Nachladen entfällt, wenn dem CCP bestimmte Programme vorgelagert sind (z. B. durch das residente Kommando EXT oder das Programm EML.COM).

Nach Einstellen wichtiger Adressen (wie beim Kaltstart) endet die Routine mit dem Aufruf von CCP.

Tastaturabfrage

Nach dieser Routine wird im Register A mitgeteilt, ob ein Zeichen von der Tastatur anliegt.

A = FFH ... Zeichen von der Tastatur liegt an

A = 0 ... es liegt kein Zeichen von der Tastatur an

Zeichen von der Tastatur übernehmen

Diese Routine übergibt im Register A ein Zeichen von der Tastatur. Liegt noch kein Zeichen an, so wird auf eine Eingabe gewartet. Das Bit 7 wird stets 0 gesetzt.

Ab der Version 1.1 erfolgt die Tastaturabfrage kontinuierlich und unabhängig von BIOS-Rufen alle 10 ms, die Tastaturwerte werden in einem 15 Byte großen Puffer abgelegt. Damit ist eine Voreintastung mehrerer Zeichen möglich.

Die vorherige und diese Routine beziehen sich somit auf den Pufferinhalt.

Ausgabe eines Zeichens auf dem Bildschirm

Das im Register C stehende Zeichen wird auf den Bildschirm ausgegeben. Das Bit 7 wird vorher auf 0 gesetzt. Enthält das Register C Steuerzeichen, so werden in der Routine entsprechende Manipulationen der Bildschirmanzeige ausgeführt.

Das Rollen des Bildschirms ist realisiert.

Ausgabe eines Zeichens auf den Drucker

Das Zeichen im Register C wird an den Drucker unter Berücksichtigung seiner Schnittstelle ausgegeben. Steuerzeichen sind zulässig, werden von der Routine aber nicht geprüft. Das Bit 7 wird hier ebenfalls auf 0 gesetzt.

Ausgabe eines Zeichens an den Lochbandstanzer

Diese Routine kann nur genutzt werden, wenn das Hilfsprogramm EML(.COM) aufgerufen worden war. Dann gibt diese Routine das im Register E stehende Zeichen an den Stanzer aus.

Lesen eines Zeichens vom Lochbandleser

Auch diese Routine ist nur nach EML(.COM) zu nutzen. Es wird ein Zeichen im 7-Bit-Code vom Leser in das Register A übergeben. Ein Paritätsbit wird nicht hinzugefügt.

Ohne EML wird im Register A stets der Wert 1AH (Endekennung) eingetragen.

Grundstellung des aktuellen Laufwerks

Es wird das mit dem nächsten Ruf ausgewählte Laufwerk in Grundstellung gebracht (durch Setzen von Kennungen).

Auswahl des Laufwerks

Für alle folgenden Diskettenoperationen wird mit dieser Routine das Laufwerk festgelegt. Es ist im Register C vorzugeben:

C = 0 ... LW A

C = 1 ... LW B usw.

Im Registerpaar HL wird die Adresse eines Parameterblockes, der das physische Laufwerk beschreibt, zurückgemeldet.

Auswahl der Spur

Im Registerpaar BC ist die Spurnummer des zu lesenden oder zu schreibenden Sektors zu übergeben.

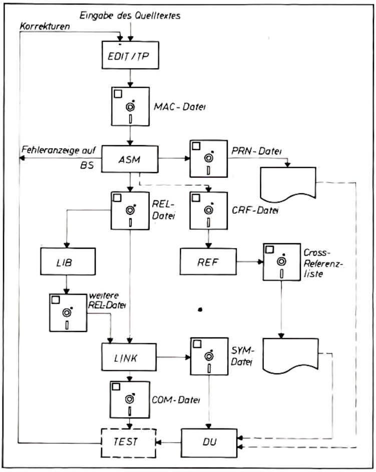


Abb.3 Überblick der Dienstprogramme

Auswahl des Sektors
Es ist in BC die Nummer des Sektors innerhalb der Spur vorzugeben. Je nach Diskettenformat sind folgende Werte möglich:
0 ... 25, 0 ... 31 oder 0 ... 63.

Vorgabe der Pufferadresse
Um einen Sektor zu lesen oder zu schreiben, ist ein 128-Byte-Puffer für die Daten im Hauptspeicher nötig. Dessen Adresse ist in BC vorzugeben und mit diesem Ruf an BIOS zu übergeben.

Lesen eines Sektors
Der mit den vorhergehenden Rufen eingestellte Sektor wird mit dieser Routine in den Puffer eingelesen.
Im Register A steht als Rückmeldung der Wert 0, wenn das Lesen fehlerfrei erfolgte, sonst eine 1.
Es werden 8 Leseversuche unternommen, bevor ein Fehler gemeldet wird.

Schreiben eines Sektors
Die 128 Byte des Puffers werden entsprechend der vorher eingestellten Diskettenparameter aufgezeichnet.
Rückmeldungen und Wiederholungen im Fehlerfall sind wie beim Lesen.

Abfrage des Druckerstatus
Von diesem Ruf wird im Register A ein FFH gemeldet, wenn der Drucker bereit ist, Zeichen zu empfangen.

Umsetzen der logischen in die physische Sektoradresse
In BC ist die logische Sektornummer, in DE die Adresse einer Übersetzungstabelle (erster Parameter des mit dem Ruf *Auswahl Laufwerk* in HL gemeldeten Blockes) vorzugeben. Dann wird durch diesen Ruf die physische Recordnummer in HL zurückgemeldet. Dies ist wichtig für Disketten mit dem Daten-Austausch-Format.

5.6.4.
Programmaufruf mit Parameterangabe

Bei einem Programmaufruf können nach dem Programmnamen von dem Bediener Parameter für das Programm eingegeben werden.
Nach Abschluß der Eingabe werden diese – mit dem Leerzeichen nach dem Programmnamen beginnend – außer im CCP-Eingabepuffer auch im Adreßbereich 81H ... FBH abgelegt; die Anzahl der Parameter wird im Byte 80H registriert. Da die ersten beiden, durch Leerzeichen getrennten Parametergruppen, oft Dateibezeichnungen darstellen, werden diese außerdem in den Standard-FCB (Adressen 5CH ... 77H) entsprechend aufbereitet übertragen. Die erste Dateibezeichnung wird ab 5CH, die zweite Dateibezeichnung ab 6CH komplett eingetragen.
Beispiel:

B>test c:datei.mac a:test.alt
Neben der Speicherung der gesamten Eingabe im CCP-Puffer ergibt sich folgende Speicherplatzbelegung:
5CH : 03H (=LW C)
5DH : 44H ("D")
5EH : 41H ("A")
5FH : 54H ("T")
60H : 45H ("E")
61H : 49H ("I")
62H : 20H
63H : 20H
64H : 20H
65H : 4DH ("M")
66H : 41H ("A")
67H : 43H ("C")
6CH : 01H (=LW A)
6DH : 54H ("T")
6EH : 45H ("E")
6FH : 53H ("S")
70H : 54H ("T")
71H : 20H
72H : 20H
73H : 20H
74H : 20H
75H : 41H ("A")
76H : 4CH ("L")
77H : 54H ("T")
Auf 80H bis 98H steht:
17H,20H,43H,3AH,44H,41H,54H,45H,49H; 2EH,
4DH,41H,43H, 20H,41H,3AH,54H,45H,53H, 54H,2EH,41H,4CH,54H,00H

5.7.
Programmentwicklungssoftware

Zur Entwicklung von Programmen im Assemblerniveau (Maschinencode) gibt es auch im Betriebssystem SCP Dienstprogramme. Sie werden als transiente Kommandos auf der Systemdiskette vertrieben.
Abb. 3 gibt einen schematischen Überblick zur Anwendung der einzelnen Dienstprogramme im Softwareentwicklungsprozeß.

5.7.1.
Erfassung des Quelltextes

Das Erfassen eines in Assemblersprache geschriebenen Programms kann mit Hilfe des Editors EDIT 1520(SCPX) oder des Textprozessors TP 1520(SCPX) erfolgen.

Der Editor ist ein zeilen- und zeichenorientierter Textaufbereiter. Mit einfachen Aufrufen lassen sich neue Programme erfassen, ändern und erweitern. Jede Zeile des Textes kann aus maximal 255 Zeichen bestehen und ist mit einer Starttaste zu beenden. Die Zeilen werden durchnumeriert und beginnen mit 100. Die Standardschrittweite ist ebenfalls 100.
Wichtige Unterkommandos des Editors sind das Einfügen, Löschen, Ersetzen und umnumerieren von Zeilen, sowie das Ändern innerhalb einer Zeile.
Mit dem Textprozessor läßt sich sehr komfortabel ein Programm in Assemblersprache erfassen, da dies bildschirmorientiert erfolgen kann. Damit ergeben sich für den Nutzer sehr einfache und wirksame Möglichkeiten, Quelltexte zu ändern, zu erweitern und zu löschen. Das Einfügen bereits erfaßter Quelltexte ist ebenso aufrufbar wie das Kopieren oder Verschieben von Teilen des Textes an andere Stellen im Text. Mit den sehr flexiblen Such- und Ersatzfunktionen wird der Textprozessor zu einem nahezu idealen Quelltexterfasser im Betriebssystem SCP.
Als Typ der entstehenden Quelltextdatei ist stets MAC zu spezifizieren.

5.7.2.
Übersetzung des Quelltextes

Die Übersetzung der mit EDIT oder dem Textprozessor erfaßten Quelltexte erfolgt mit dem Assembler ASM 1520(SCPX).
Dieser Assembler übersetzt alle CPU-Befehle des U 880, also auch die für die E/A-Arbeit wichtigen BDOS-Funktionen, und er verarbeitet eine große Anzahl von Pseudobefehlen mit verschiedensten Wirkungen.
Die Leistungsfähigkeit des Assemblers wird unter anderem von folgenden Eigenschaften bestimmt:

- Einzelfunktions-Pseudooperationen
– zur Daten- und Symboldefinition
– zur Speicherplatzverwaltung
– zur Format- und Listensteuerung von Ausdrucken
– zum Einfügen anderer Quelltexte, u. a.
- Makrofähigkeit

Damit kann ein Block von Anweisungen, der einmal erfaßt wurde, mehrfach in die entstehende Datei eingefügt werden.

- Bedingte Assemblierung

Es können Bedingungen im Quelltext vorgegeben werden, unter denen dann Teile des Quelltextes übersetzt oder übergangen werden.
Der Assembler belegt einschließlich notwendiger Puffer 23 KByte im Hauptspeicher.
Je nach Aufruf des ASM können verschiedene Dateien beim Übersetzen entstehen, ein reiner Fehlersuchlauf ist auch möglich.
Im Standardfall entstehen eine Objektdatei vom Typ REL und eine Druckdatei vom Typ PRN. Festgestellte Fehler werden auf dem Bildschirm ausgewiesen.
Mit zusätzlichen Eingaben – Schalter genannt – läßt sich auch eine spezielle Datei (Typ CRF) erzeugen, aus der mit dem transienten Kommando REF eine umfangreichere Druckdatei, die Cross-Referenz-Liste, erzeugt werden kann.
Die Objektdateien lassen sich mit dem transienten Kommando LIB 1520(SCPX) zu Bibliotheken zusammenfassen.

5.7.3.
Binden der Objektprogramme

Mit dem Programmverbinder LINK 1520(SCPX) werden einzelne Objektprogramme (REL-Dateien), die sich auch in einer mit LIB gebildeten Bibliothek befinden können, zu einem ausführbaren Programm im Maschinencode (COM-Datei) gewandelt.
Neben der Angabe der zu bindenden REL-Dateien und des Namens der COM-Datei können wieder Schalter vom Bediener eingegeben und so besondere Funktionen ausgelöst werden. So ist zum Beispiel das Ablegen der COM-Datei auf der Diskette, das Durchsuchen bestimmter Bibliotheken und das sofortige Ausführen des Programms nach dem Bindelauf möglich.
Für den folgenden Test mit dem Debugger läßt sich auch mit einem Schalter eine spezielle Datei, die SYM-Datei, erzeugen. Der Linker belegt einen Speicherbereich von 16 KByte.

5.7.4.
Testen des Programms

Für den Test eines assemblierten und gebundenen Programms ist der Debugger DU 1520(SCPX) vorgesehen. Dieses transiente Kommando realisiert mit zahlreichen Unterkommandos wichtige Funktionen, unter anderem:

- Starten eines Programms im Echtzeitbetrieb mit Angabe einer Startadresse und zweier Haltepunkte
- Schrittweises Abarbeiten eines Programms mit Protokollieren der CPU-Register auf dem Bildschirm
- Anzeige von Speicherbereichen
- Füllen von Speicherbereichen
- Verschieben von Speicherbereichen
- Anzeige und Ändern einzelner Speicherplätze.

Als Adressen lassen sich auch symbolische Adressen aus dem Quelltext angeben, wenn beim Bindelauf die SYM-Datei erzeugt worden war.
Der Debugger mit einer Größe von etwa 10 KByte wird nach seinem Aufruf bis an die obere Grenze von TPA verschoben, um so den Speicherbereich ab 100H für das zu testende Programm freizugeben.

edv aspekte

lieferbar
2/85

CAD/CAM

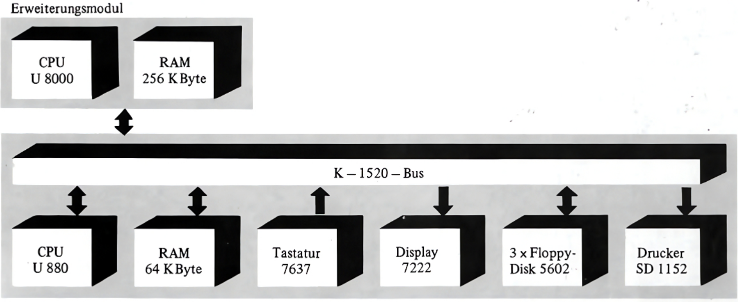
Entwicklung und Anwendung

Mit einer Zusammenstellung von Beiträgen zum rechnergestützten Konstruieren und Fertigen wird der zunehmenden Automatisierung von Produktions- und anderen Prozessen unserer Volkswirtschaft Rechnung getragen.

Bestellungen richten Sie bitte an den Verlag Die Wirtschaft, Abt. Vertrieb, 1055 Berlin, Am Friedrichshain 22.

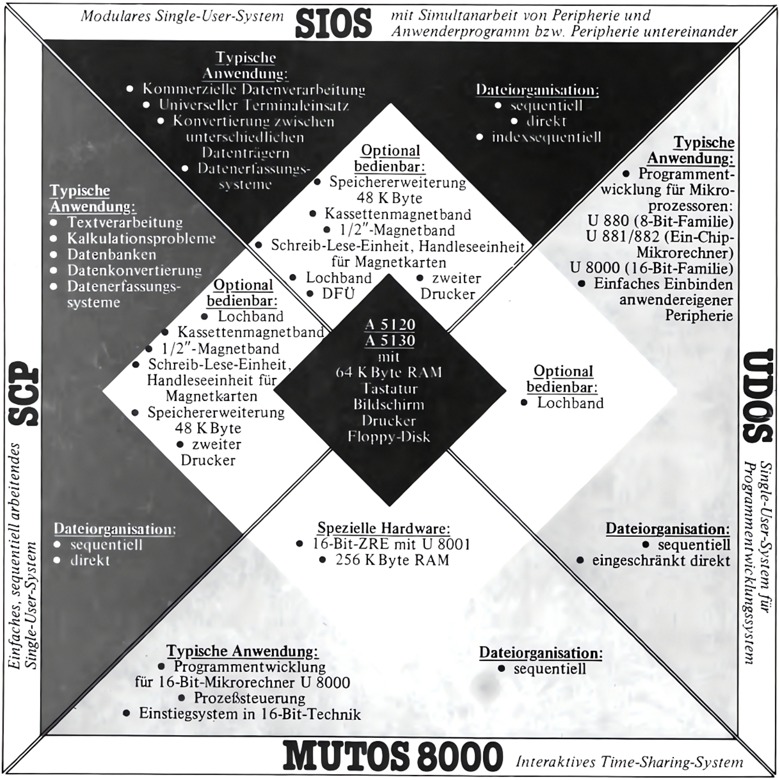
Erweiterter Bürocomputer robotron A 5120.16

Seit 1985 wird vom VEB Kombinat Robotron der erste serienmäßig produzierte 16-Bit-Bürocomputer der DDR – der A 5120.16 – angeboten. Basis ist der bekannte A 5120, erweitert um einen zusätzlichen Hardwaremodul und mit dem entsprechenden Betriebssystem versehen. Der Erweiterungsmodul besteht aus einer Leiterkarte mit dem 16-Bit-Mikroprozessor U 8001 und einer Speicherkarte mit bis zu 4×64 = 256 KByte RAM, die gleichzeitig einen Kommunikationskanal zwischen 8- und 16-Bit-Rechner auf der Basis des PIO U 855 realisieren.
Der A 5120.16 läßt sich sowohl als 8-Bit- als auch als 16-Bit-Mikrorechner betreiben. Das heißt, vorhandene Anwendersoftware für den A 5120 unter den Betriebssystemen SIOS, UDOS oder SCP kann weiterhin voll genutzt werden.
Beim Einsatz als 16-Bit-Rechner dient der 8-Bit-Prozessor des A 5120 als Ein-/Ausgabeprozessor für die gesamte Rechnerperipherie.
Für den A 5120.16 wurde aus dem Betriebssystem MUTOS 1600 des Basisrechners robotron A 6402 das Betriebssystem MUTOS 8000 entwickelt, welches zu UNIX und zum System INMOS für SKR-Mikrorechner kompatibel ist. Die ausführliche Beschreibung von MUTOS 8000 finden Sie in diesem Heft auf den Seiten 5 bis 16.



aspekte **blickpunkt**

Welches System wofür?



ISSN 0232-6833 · edv-aspekte · Berlin · 5 (1986) 1 · S. 1–64